TFG de Grado de Ingeniería Electrónica Industrial y Automática

PLATAFORMA DE CHECK-IN DINÁMICO MEDIANTE TECNOLOGÍA NFC EN SMARTPHONES

Autor: David Lagoa Quintana

Tutor: Alejandro Iván Martín Clemente

Agradecimientos.

A mi familia y amigos por todo su apoyo y ayuda a lo largo de todos estos años de esfuerzo y sacrificio.

Gracias.

Resumen.

La utilización de los Smartphone está tomando un mayor protagonismo cada día, por ello es una herramienta casi indispensable en muchos ámbitos. Dada las grandes posibilidades de desarrollo de aplicaciones que ofrecen los Smartphone, podremos utilizar estos dispositivos como ayuda en el día a día. En concreto en este proyecto se utilizará la tecnología NFC presente en algunos de estos Smartphone junto con el acceso a una base de datos externa.

Los usos que permite la tecnología NFC en estos dispositivos son varios, en concreto en este proyecto se utilizará junto con etiquetas NFC, las cuales son elementos pasivos cuya utilización es posible gracias a la energía procedente del Smartphone. Estos dos elementos se emplearan para realizar un control de accesos de los usuarios a un determinado edificio.

Para llevar a cabo este proyecto se ha diseñado una aplicación en la plataforma Android que permite gestionar una base de datos alojada en un servidor, además de leer y escribir información en un tag o etiqueta NFC.

Con todos estos elementos se ha coordinado un sistema simple y eficaz de consultar y manejar los datos de una base de datos externa. Para esta solución, se emplea el uso de las etiquetas NFC para guardar un código de usuario, que el sistema es capaz de interpretar y representar junto con todos los datos del usuario asignados a este código.

Abstract.

The use of Smartphones is taking a greater role every day, it is an indispensable tool in many areas. Due to the high potential for application development, we can use these devices day by day helping us in our affairs. In this project will be used NFC technology, present in many Smartphones, with the access to an external database.

The uses that allow NFC technology to these devices are many, particularly in this project will be uses with NFG tags, which are passive devices that works with the energy produced by the Smartphone. These two elements were used to do an access control for users into a building.

In order to develop this project has been designed an application on Android platform to manage a database hosted on a server. In addition the application is capable to read and write information on a NFC tag.

With all these elements has been coordinated a simple and effective system that allows querying and managing data store in an external database. In these case, NFC tags are used to store a user code. The system is able to interpret and show all the information of an user with the identified code.

Índice de contenido.

1 Introducción.

Actualmente en el día a día una persona tiene que atravesar numerosas puertas, torniquetes, barreras… ya sea en el hogar, en el transporte público, lugar de estudios, trabajo, etc. Uno de los problemas que esto implica es que se necesitan llaves, permisos, billetes para el transporte o cualquier elemento que identifique a esta persona como capaz de superar esa barrera.

El problema que supone el llevar tantos elementos es la posible pérdida u olvido de uno de ellos. De este modo se ha pensado en un sistema capaz de identificar a un usuario de una manera simple y rápida, con la utilización de un simple elemento. Uno de estos elemento sería el Smartphone que hoy en día es altamente utilizado y el otro elemento sería un tag o etiqueta NFC capaz de ser integrada con gran facilidad, por ejemplo, en una tarjeta bancaria.

En este proyecto se diseña un sistema capaz de controlar un sistema de control de accesos con los dos elementos indicados anteriormente y aprovechando la característica de un Smartphone para conectarse a Internet y que este pueda consultar una base datos alojada en un servidor Web.

Para que este sistema sea eficaz ha sido necesaria la introducción de una base de datos externa para que de este modo la información de identificación de usuarios pueda ser obtenida en distintos dispositivos móviles, ya que de lo contrario la base de datos quedaría guardada en un sólo dispositivo y la aplicación perdería mucha funcionalidad.

Los objetivos de este trabajo son el estudio de las bases de datos, la tecnología NFC y el desarrollo en Android para su aplicación en el desarrollo de un control de accesos.

Para llevar a cabo este proyecto se han elegido diferentes entornos de programación que responden a los requisitos que la aplicación creada ha requerido.

2 Motivación.

La principal motivación que ha dado lugar a este proyecto ha sido el interés por conocer cómo funcionan los dispositivos Android que diariamente se utilizan por una gran variedad de usuarios. De este modo he tenido la oportunidad de aprender un nuevos lenguajes de programación como el lenguaje java, lenguaje XML y SQL.

La plataforma elegida para el desarrollo de este trabajo ha sido Android, ya que esta plataforma de desarrollo permite utilizar todas las capacidades de los dispositivos de una manera libre y gratuita. De este modo, durante el proceso de desarrollo de este estudio, otra motivación encontrada es la infinidad de posibilidades que permite Android a la hora de trasladar las ideas a aplicaciones para todo tipo de dispositivos tan manejables como un teléfono móvil o tableta, elementos que hoy en día están disponibles y al alcance de cualquier persona.

También una motivación encontrada es la capacidad de ampliación que tiene este trabajo y el saber que las tecnologías empleadas tienen grandes posibilidades de futuro.

3 Objetivos.

El objetivo de este proyecto es diseñar un sistema capaz de controlar los accesos a un determinado edificio, con la ayuda de una base de datos externa controlada por una aplicación para Smartphone.

Los objetivos de este trabajo son el estudio del trabajo con las bases de datos, el desarrollo de aplicaciones para Smartphone y el empleo de etiquetas NFC.

Una vez estudiadas las posibilidades de los tres elementos anteriores se diseñara un sistema que usa la tecnología NFC presente en los algunos dispositivos móviles, que al ser utilizada junto con el empleo de etiquetas NFC posibilita crear un sistema de reconocimiento de identidades y accesos. Para alojar las diferentes identidades se utiliza una base de datos alojada en un servidor.

4 Estado del arte.

Los elementos que participan en este proyecto son los teléfonos móviles, el acceso a bases de datos a través de internet y la tecnología NFC cuyo antecedente era conocido como RFID.

4.1 Telefonía móvil.

Un teléfono móvil es un dispositivo capaz de recibir llamadas telefónicas gracias a una conexión de radio a lo largo de un área geográfica amplia[1]. Esto es posible gracias a que estos dispositivos se conectan a una red móvil cuyos proveedores son los operadores de telefonía móvil, los cuales permiten acceso a la red pública de teléfono.

Adicional a la telefonía, los teléfonos móviles soportan una gran variedad de servicios como el envío de mensajes de texto, MMS, conexión a internet, comunicaciones infrarrojos, bluetooth, juegos, realización de fotos y videos, etc. Los teléfonos que ofrecen estas y más capacidades de computación se conocen como los smartphones.

Los primeros dispositivos de telefonía se empezaron a desarrollar a finales de los años 40 en Estados Unidos, en un principio estos teléfonos utilizaban la modulación AM y posteriormente la modulación FM, estos equipos eran enormes y pesados por lo que se transportaban y eran instalados en vehículos.

No fue hasta 1973 cuando la compañía Motorola creo el primer teléfono móvil de mano, este pesaba 1.1 kg y medía 23 cm de largo, 13 cm de profundidad y 4.45 cm de ancho. Su batería duraba 30 minutos de habla y para su carga se necesitaban 10 horas.

Poco a poco se fue desarrollando la telefonía y en 1981 el fabricante Ericsson fabricó un teléfono que utilizaba canales de radio analógico (frecuencias de 450 MHz) con una modulación FM. Ericsson mejoro el sistema llevando a una frecuencia de 900 MHz en 1986. Esta generación de teléfonos fue conocida como primera generación (1G).

Durante la década de 1990 apareció la segunda generación o 2G que utiliza sistemas como el GSM, IS-136, iDEN e IS-95. Durante esta generación se trata de desarrollar la digitalización de las comunicaciones. El estándar que ha universalizado la telefonía móvil ha sido el GSM (Global System for Mobile), aunque estaba dotado de buena calidad de voz, itinerancia, terminales de reducido peso y tamaño, fue quedándose obsoleto debido a que los terminales cada día estaban más desarrollados y permitían un aumento de la capacidad de transferencia de datos. Debido a esto la velocidad del GSM no era lo suficiente para que estos nuevos terminales fueran útiles.

La siguiente evolución de la telefonía fue el 2.5G que permitía el servicio de mensajería EMS y MMS. Durante este periodo se utilizaban la red GPRS (General Packet Radio Service) que permite velocidades de entre 56-114 kbits/s y de la tecnología EDGE (Enhanced Data rates for GSM Evolution) que tenía una velocidad de datos de hasta 384 kbit/s.

La tercera generación 3G llega de la necesidad de aumentar la capacidad de transmisión de datos para poder ofrecer una conexión a internet móvil, televisión o descarga de archivos. En esta generación el desarrollo tecnológico permite el uso de un sistema totalmente nuevo. UMTS (Universal Mobile Telecommunications System) que utiliza la tecnología CDMA que permite obtener velocidades de transmisión de datos de hasta 7.2 Mbit/s.

Por último en la actualidad se está instaurando la cuarta generación o 4G que tiene un ancho de banda mucho mayor y permite velocidades de transferencia muy elevadas dependiendo la infraestructura de la que disponga.

4.2 Smartphone.

Un Smartphone es un dispositivo móvil de última generación que proporciona un procesado y una administración de hardware y software similar a la de un ordenador[2].

4.2.1 Historia.

Las primeras combinaciones de teléfonos y procesado fueron conceptuados en 1973 pero no fue hasta los principios del año 1993 cuando se empezaron a comercializar.

El primer teléfono móvil en incorporar las características de una PDA fue un prototipo de IBM en el año 1993. En los finales de los noventa era común tener dos dispositivos por separado, un teléfono móvil y una PDA. Aparecieron dispositivos que combinaban ambas funciones como el NOKIA 9000, el Ericsson R380 o el Kyocera 6035 con sus diferentes sistemas operativos.

Actualmente los sistemas operativos más habituales en el que se desarrollan los Smartphone son los sistemas Android, iOS, Windows Phone o Blackberry. Previamente a estos sistemas operativos el sistema operativo más usado hasta el 2010 era Symbian.

4.2.2 Android.

Android es una plataforma de código abierto fundada en 2003 por Andy Rubyn y con el respaldo de Google, junto con los principales fabricantes de hardware y software (HTC, Intel, ARM, Motorola y Samsung) que forman la "Open Handset Alliance" (OHA)[3] que es una alianza comercial de compañías dedicada a desarrollar estándares abiertos para dispositivos móviles.

En 2008 apareció el HTC Dream que era el primer Smartphone que usaba Android. Este utiliza aplicaciones propiedad de Google así como también tiene la posibilidad de la integración de aplicaciones de terceras partes que están disponibles en Google Play, que en octubre de 2008 fue lanzado como Android Market. En 2010 Android se convirtió en la plataforma de Smartphone más vendida.

Para el desarrollo de aplicaciones en este sistema operativo, al ser una plataforma de código abierto Android pone a la disposición de los usuarios los recursos necesarios para desarrollar aplicaciones para el uso en este sistema operativo[4].

4.2.3 iOS

iOS es el sistema operativo desarrollado y distribuido por Apple Inc. Apple reveló la existencia de iPhone OS en el 2007 aunque el sistema no tomo este nombre como oficial hasta el 2008. Fue en 2010 con el lanzamiento del iPhone 4 cuando se dio a conocer el actual nombre del sistema operativo iOS[5].

Desde el 2008 la compañía Apple Inc. pone a la disposición de sus sistemas operativos iOS un servicio conocido por App Store que permite descargar aplicaciones propias y de terceros.

Durante el año 2008 el kit de desarrollo de software (SDK) fue liberado para el desarrollo de aplicaciones por terceros. Este sistema operativo requiere el pago de la cuota del "iPhone Developer Program" para poder desarrollar aplicaciones para dispositivos con este sistema operativo, al contrario que otras plataformas cuya plataforma es de código abierto.

4.2.4 Windows Phone

Windows Phone es un sistema operativo móvil que apareció en 2010 y desarrollado por Microsoft como sucesor de Windows Mobile. A diferencia de su predecesor Windows Phone está enfocado al mercado de consumo y no al mercado empresarial[6].

La tienda de Windows Phone es usada para la distribución de aplicaciones. Está gestionada por Microsoft que se encarga de probar cada una de las aplicaciones que son publicadas, de modo que filtra las aplicaciones que incluyan contenidos no aptos para todos los públicos.

El desarrollo de aplicación para este sistema operativo se puede realizar de dos maneras.

- Mediante Microsoft Silverlight indicado para el desarrollo de aplicaciones basadas en XAML.
- Mediante Microsoft XNA Framework indicado para el desarrollo de videojuegos.

4.2.5 BlackBerry

En 1999 la compañía RIM lanzo el primer equipo BlackBerry, el cual permitía un sistema de comunicación de correo electrónico seguro en tiempo real en sistemas móviles. Servicios como BlackBerry Messenger permiten la integración de todas las comunicaciones en una simple entrada. Recientemente BlackBerry está creando equipos en la nueva plataforma llamada “BlackBerry 10”[7].

BlackBerry World es la tienda virtual para el sistema operativo BlackBerry el cual permite instalar en sus dispositivos las aplicaciones disponibles. También pueden instalar aplicaciones de Android en los últimos dispositivos BlackBerry a través de Google Play Store de Android.

4.3 Tecnología NFC.

NFC (Near Field Communication) es una tecnología inalámbrica de corto alcance. El alcance de esta comunicación, en principio, no debe ser mayor de 10 o 15 cm[8].

El NFC es una evolución del estándar RFID (Radio Frequency IDentification), funciona bajo el estándar RFID. El NFC es la unión del RFID y las tecnologías interconectadas. Funciona sobre una frecuencia libre, lo cual implica que no necesita de ninguna licencia especial para su utilización.

Es una tecnología destinada a la comunicación instantánea, su tasa de transferencia es de hasta 424 kbit/s, es una tecnología útil para la identificación, validación de equipos/personas, en el ámbito publicitario, para la realización de pagos, etc…

Esta tecnología es compatible con el estándar ISO/IEC-14443 de tarjetas de proximidad sin contacto, por lo que esto lo hace compatible con toda la infraestructura existente de pago sin contacto y transporte que actualmente está en uso.

4.3.1 Orígenes.

En el 2002 Philips y Sony intentaron conseguir un protocolo compatible con las tecnologías que existían en este momento, FeliCa de Sony y Mifare de Philips.

En 2003 se aprobó como estándar el NFC en la normativa ISO 18092.

Ya en 2004 Philips, Sony y Nokia crearon NFC Forum [8] abriendo la posibilidad de que empresas como Visa, Google, PayPal, At&t pertenezcan a él y apoyen la tecnología NFC.

Algunos miembros de esta organización son empresas como: Google, Visa, Dell, Intel, Microsoft, Samsung, Sony, At&t, Paypal, American Express, Barclaycard, BlackBerry Limited, Cambridge Silicon Radio, Canon INC., Hewlett Packard, Huawei Technologies Co.Ltd., Nokia(uno de los mayores impulsores de la tecnología) al igual que otras muchas empresas de gran reputación.

Las funciones o propósitos de NFC Forum son entre otros:

- Desarrollar las especificaciones de estándares.
- Crear las arquitecturas e incluso los parámetros de interoperabilidad para el protocolo.
- Desarrollar los dispositivos NFC.
- Enseñar a las empresas y a los consumidores como comprender la tecnología NFC.

No fue hasta el 2006 cuanto se expuso la primera especificación técnica de esta tecnología.

4.3.2 Funcionamiento.

Para poder utilizar esta tecnología es necesario disponer de al menos un dispositivo que soporte la tecnología NFC y un tag, o etiqueta, RFID en la que se encuentra almacenada la informacion. También es posible la comunicación entre dispositivos que soporte esta tecnología. Lo necesario es que haya un dispositivo que inicie esta comunicación y sea capaz de conocer el estado de esta comunicación.

Para poder usar esta tecnología simplemente tenemos que aproximar un dispositivo con tecnología NFC a un lector, etiqueta u otro dispositivo con la tecnología. A la hora de acercar los dos elementos, el lector del dispositivo emite una señal de radio de corto alcance que es capaz de excitar la bobina/microchip de la etiqueta, con lo que es capaz de obtener la información contenida en este dispositivo.

Hay dos tipos de funcionamientos posibles en esta tecnología.

- Activo, en este caso tanto el emisor como el receptor de la información generan señal de radio o campo electromagnético. Este podría ser el caso de dos dispositivos NFC (smartphones).
- Pasivo, en este caso solo uno de los elementos implicados en la comunicación genera campo electromagnético. El dispositivo que inicia la comunicación es el que genera el campo electromagnético y el otro hace uso de esta señal de radio para transferir la información contenida en este. Como ejemplo podríamos indicar el caso en el que se realiza la comunicación con etiquetas.

Existen varios modos de comunicación:

- Punto a punto, utilizado para el intercambio de datos o establecimiento de comunicaciones entre dispositivos NFC. Como ejemplo de este modo estaría la vinculación de dispositivos, o el intercambio de archivos.
- Lectura/escritura, este modo se utiliza para introducir o recibir datos de un tag o etiqueta de manera pasiva. En este caso como ejemplo podríamos utilizar esta tecnología para escribir una identificación en una etiqueta y posteriormente realizar su lectura.
- Emulación de tarjeta, el dispositivo NFC envía informacion que puede interpretar un lector. Un ejemplo práctico utilizado en la actualidad es el uso del dispositivo como forma de pago al acercarlo a un dispositivo lector.

4.3.3 Ventajas.

Entre las principales ventajas del uso del NFC encontramos las siguientes:

- Alta velocidad de establecimiento, al no necesitar emparejamiento el establecimiento de la comunicación es prácticamente instantánea, solo hace falta acercar los dispositivos.
- Fácil acceso a los datos.
- Operabilidad y versatilidad, tiene multitud de usos y al tener accesos a la tecnología de nuestros smartphone este es capaz de interpretar toda la información y no es necesario de otro dispositivo.
- Alto grado de seguridad, dado a su corto alcance.

4.3.4 Especificaciones técnicas.

Uno de los factores más importantes de la tecnología NFC son sus estándares, esta tecnología sigue las normas ISO/IEC 14443 e ISO/IEC 18000-3[9].

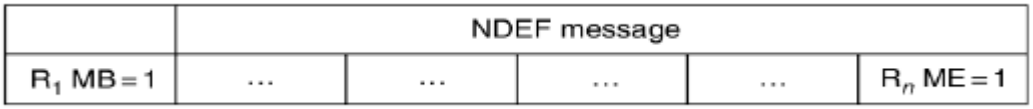
- En cuanto a la norma ISO/IEC 14443, esta define las tarjetas de identificación que se utilizan para almacenar la información, como la que se encuentra en las etiquetas de NFC
- La norma ISO/IEC 19000-3 es un estándar internacional que define que todos los dispositivos se comunican con una frecuencia de 13,54 Mhz, además los dispositivos deben estar a una distancian de 4 cm para que se pueda realizar el traspaso de datos.

Esta tecnología tiene cinco principales especificaciones las cuales han sido definidas por el NFC Forum, a continuación se exponen estas especificaciones que forman parte de este protocolo de comunicación:

- NDEF (NFC Data Exchange Format)

  Esta característica define el formato de intercambio de información pudiendo estar formado por uno o más registros, complementando así un mensaje NDEF.

  A continuación vemos la estructura que tiene un mensaje NDEF:



  En el primer registro de un mensaje NDEF se encuentra la cláusula Begin (flag MB = 1) y el último de los registros contiene el mensaje END (ME = 1) por lo tanto, MB y ME indican el inicio y el final respectivamente del registro.

  Como los datos se guardan en el registro NDEF pueden ser de diferentes tipos y tamaños, cada dato enviado tiene asociado tres atributos: la duración, el tipo y un identificador.

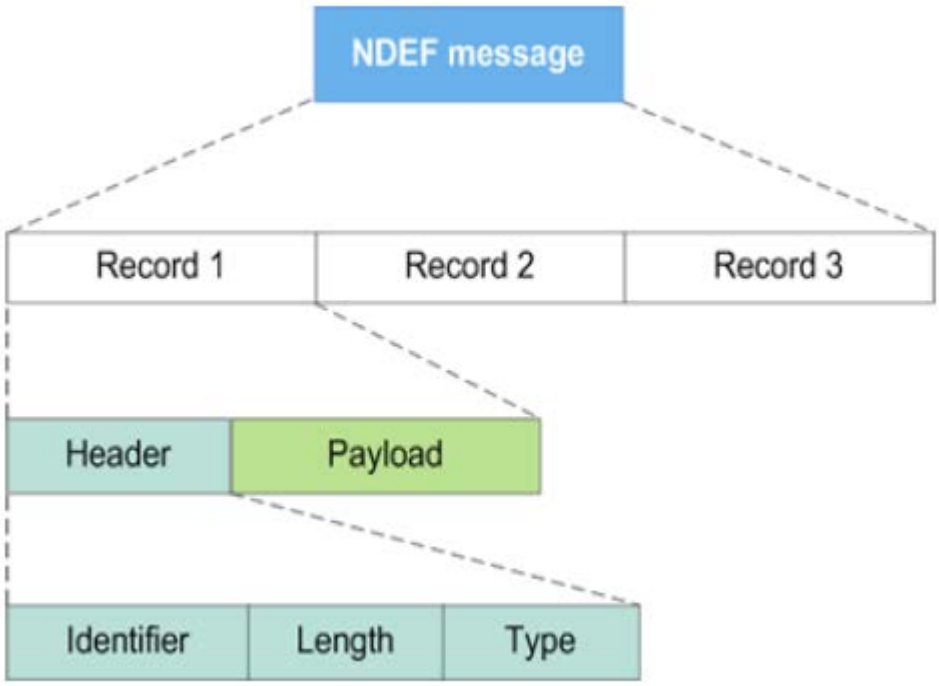
- NFC Record Type Definition (registro NDEF)

  La especificación NDEF expone el formato común para la información pero no especifica ninguno en detalle, por lo tanto estas características van por separado.

  Los diferentes tipos que puede contener cada registro NDEF se guardan en una cadena con el nombre RTN (Records Type Name) en la que se pueden especificar diferentes formatos como: MIME Types, URIs u otros tipos NFC conocidos.
  Los tipos de formato conocidos por NFC Forum contienen un identificador propio (NID) no se puede utilizar este identificadores para usar otro tipo de formato no conocido por NFC Forum. Los hay de dos tipos, locales y globales.

Los tipos externos a NFC Forum son los usados por organizaciones que requieren utilizar un formato propio para sus fines. Estos tipos de formato tienen otro tipo de identificador.

Los mensajes NDEF tienen la siguiente estructura:



- NFC Forum URI Record Type (RTD)

RTD especifica el formato y la reglar para crear un tipo estándar usado por aplicaciones definidas por NFC Forum y por terceras partes que están basadas en el formato NDEF. La especificación RTD aporta una manera eficiente de definir tipos de formato para aplicaciones y da al usuario la oportunidad de crear sus propias aplicaciones basadas en especificaciones de NFC Forum.

De este modo podemos utilizar los tipos de formato como NFC Text RTD o NFC URI RTD de una manera eficiente.

- Text Record

Es de texto y puede ser usado en combinación con otros campos para aportar un extra de información al contenido del tag/etiqueta.

El texto de esta etiqueta puede ser sobrescrito por agentes externo y debido a esto es recomendable solo usarlo para propósitos informativos y no emplearlo para ningún otro tipo de acción.

- Smart Poster

El tipo Smart Poster sirve para incluir metadatos a la URI. Una etiqueta de este tipo permite desencadenar acciones a realizar en el dispositivo que lee esta información, como abrir el navegador en una determinada URL.

Los datos que son enviados a trabas del mensaje NDEF Smart Poster es un conjunto de múltiples mensajes de registros NDEF.

Los tag/etiquetas del tipo Smart Poster pueden llevar uno o más de los siguientes componentes:

- Campo Title: este registro es una instancia de Text_RTD, es opcional y puede ser varias veces usado

- Campo URI: es la extensión del campo URI, este campo añade metadatos a la URI forma parte del núcleo del registro de la etiqueta y no puede ser repetido (una sola URI por campo de Smart Poster)

- Campo Action: es optativo y es el encargado de desencadenar las acciones, como abrir una determinada aplicación, abrir el navegador, etc

- Campo Icon: este parámetro también es opcional y es utilizador para incluir una o más imágenes de tipo MIME de modo que el dispositivo que tome la información de esta etiqueta las guarde.

- Campo Size: parámetro opcional que indica el tamaño del contenido. Este parámetro es más útil si se usa en combinación con el parámetro que indica el tipo, para que de este modo determinar si el dispositivo puede o no representar la información de la etiqueta.

- Campo Type: también es de carácter opcional y es usado para determinar si el dispositivo está capacitado para acceder a la información de este objeto externo.

4.3.5 Seguridad.

Esta tecnología no está exenta de ataques, la mayoría de estos ataques usan la principal característica del NFC, su usabilidad o transparencia que ofrece el usuario.

Uno de los posibles ataques es el conocido Relay-Atack, que consiste en dirigir la información de un dispositivo a otro. La forma de evitar este tipo de ataques es realizando una redundancia en la comprobación, es decir, autentificar dos veces a la hora de realizar las transacciones.

Otro ataque posible que podría suceder es la puesta en marcha de servicios del dispositivo sin consentimiento del usuario, como la activación del bluetooth y su uso para acceder a ficheros del dispositivo atacado.

Otro tipo de ataque conocidos como “Modificación”, “Corrupción”, “Man in the middle”, “Denegación de servicio” o “Snnifing” son difíciles de llevar a cabo debido a que esta tecnología requiere la proximidad para poder realizar operaciones.

En cuanto al pago mediante NFC, su seguridad está basada del mismo modo que en la seguridad que pueda tener una tarjeta de crédito. Debido a que cuando estamos en un comercio ante todo confiamos en que la transacción es fiable.

Otro posible problema que puede surgir es que no todas las aplicaciones cifran su informacion, de modo que esto haría posible que el atacante se hiciera con el control del teléfono y por lo tanto del contenido que se encuentra en este.

Actualmente se está trabajando en tener una seguridad añadida incluyendo un Chip NFC. Como ejemplo la compañía Samsung incluye un chip “Secu-NFC” que se encarga de almacenar la información más sensible a ataques, como datos de pago, contraseñas y cifrado. Una de las empresas más importantes en la creación de semiconductores como es NXP también está integrando chips de seguridad NFC en numerosos dispositivos.

4.3.6 Tipos de etiquetas.

Los tags / etiquetas NFC son dispositivos físicos donde se guarda la información, las hay de diferentes tipos y formas. Hay cuatro tipos empleados por esta tecnología:

- Tipo 1: Esta etiqueta está basada en la normativa ISO 14443-A. Estas etiquetas son de lectura y escrituras y los usuarios pueden configurar las etiquetas para ser sólo de lectura.

- Tipo 2: basadas en la normativa ISO 14443-A. Estas etiquetas son de lectura y escritura. Las hay disponibles en varios formatos, etiquetas de interior y etiquetas sin contacto. Se ha utilizado en varias RFID y soluciones RFID/NFC.

- Tipo 3: Esta etiqueta incluye un chip sin contacto IC. Está basada originalmente en la Norma Industrial Japonesa (JIS) X 6319-4, que también se conoce como FeliCa. En este caso las etiquetas están preconfiguradas en fábrica para ser de escritura o sólo de lectura. La memoria de estas etiquetas es variable.

- Tipo 4: Estas etiquetas son compatibles con la norma ISO / IEC 7816-4 y con la norma ISO 14443-A e ISO 14443-B. Estas etiquetas se preconfiguran en fabrica para ser leído y pueden ser de re-escritura o sólo de lectura.

4.4 Bases de datos.

Una base de datos es un conjunto de datos organizado. Los datos son típicamente organizados de tal modo que se vea reflejado los aspectos importantes de la realidad de tal modo que sea fácil la comprensión de la misma[10].

Los sistemas de manipulación de bases de datos (DBMSs) son aplicaciones diseñadas específicamente para interactuar con el usuario, otras aplicaciones y con la base de datos para poder obtener informacion y analizar datos. El propósito general de un DBMS es un software diseñado que sea capaz de permitir la definición, creación, consulta, actualización y administración de bases de datos. Algunos DBMSs conocidos serían MySQL, MariaDB, PostgreSQL, Micrisoft SQL Server, Oracle, SAP, dBASE, FoxPro, IBM DB2, LibreOffice Base y FileMaker Pro. Habitualmente no es posible que diferentes DBMSs puedan ser compatibles los unos con loso otros, pero existen algunas compatibilidades gracias a unos estándares entre algunos de estos DBMSs.

4.4.1 Historia.

Con el progreso de la tecnología en áreas de procesadores, memorias y conexiones de los ordenadores, los tamaños, capacidades y respuesta de las bases de datos y sus respectivos DBMSs se han desarrollado.

El desarrollo de la tecnología en las bases de datos en función del modelo o estructura: navigacional, SLQ/relacional y post-relacional. Los dos primeros principales modelos navigacionales de datos fueron modelos jerárquicos como el sistema IMS de IBM y el model Codasyl. El modelo relacional, propuesto primeramente en 1970 por Edgar F. Codd, partía de la tradición de que las aplicaciones debían buscar el dato por el contenido, en lugar de seguir el link a este dato. El sistema relacional está formado por tablas con las que se definen las relaciones. Sin embargo, no fue hasta mediados de los años 80 cuando el hardware disponible fue suficientemente poderoso como para poder trabajar con sistemas relacionales (aplicaciones DBMSs). A principios de los 90 era el modelo relacional era el que dominante en las aplicaciones de procesado de datos. El lenguaje dominante en las bases de datos es el estándar SQL del modelo relacional, que fue influenciado por lenguajes de bases de datos de otros modelos[11].

Las bases de datos orientadas a objetos fueron inventadas en los años 80 para resolver el problema que existía a la hora de utilizar un modelo relacional con un programa escrito con un lenguaje de programación orientado a objetos, lo que condujo a la acuñación del término “post-relacional” y también al desarrollo de modelos híbridos relacionales y orientados a objetos[12].

La siguiente generación de bases de datos post-relacionales, a partir del año 2000, comenzaron a llamarse "NoSQL", en la que se introdujeron bases de datos orientadas a documentos y atajos. Al mismo tiempo la siguiente generación conocida como "NewSQL" trataba de implementar el modelo relacional junto con las altas prestaciones de NoSQL.

5 Android.

Android funciona en cientos de millones de dispositivos móviles en 190 países de todo el mundo. Es el sistema operativo más instalado en plataformas móviles y sigue creciendo. Cada día otro millón de usuarios hace funcionar su dispositivo Android por primera vez y comienzan la búsqueda de aplicaciones, juegos u otros contenidos digitales[13].

Android aporta una plataforma para la creación de aplicaciones y juegos para este sistema operativo, como también una tienda virtual abierta para distribuir estas aplicaciones instantáneamente.

5.1 Alianzas globales.

Este sistema operativo ha sido creado gracias a las contribuciones de la comunidad de código abierto Linux y de más de 300 socios desarrolladores de hardware y software, convirtiéndose Android así en el sistema operativo móvil con el índice de desarrollo más rápido.

La creación de Android se ha convertido en el favorito de desarrolladores y consumidores generando así un crecimiento del consumo de aplicaciones. Los usuarios de Android descargan en torno a un billón y medio de aplicaciones y juegos de Google Play cada mes.

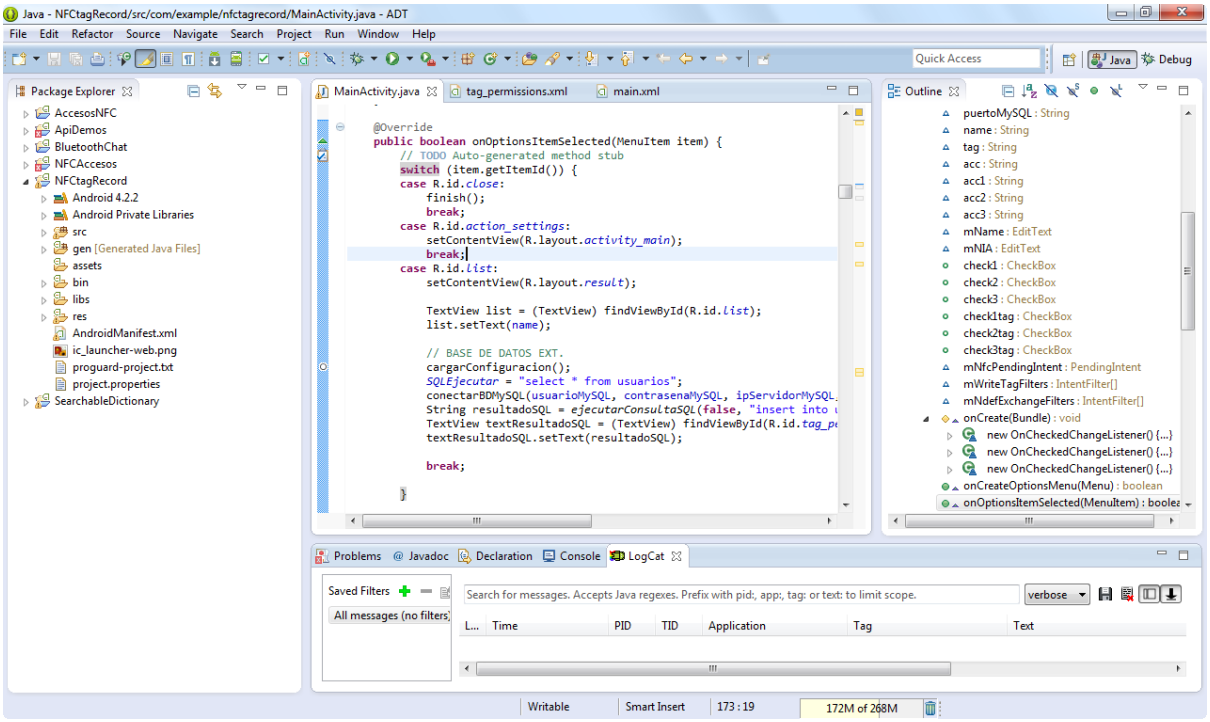
Continuamente, Android está desarrollando junto con sus socios mejoras de hardware y software de modo que tanto usuarios y desarrolladores pueden aprovechar las nuevas capacidades que se van incluyendo.

5.2 Entorno de desarrollo.

Android pone a la disposición de los usuarios todo lo necesario para crear de la mejor manera las aplicaciones. Para crear aplicaciones en Android es necesario sólo una aplicación que permite desarrollar y simular las aplicaciones creadas para teléfonos o tablets.

Android también pone a la disposición de los desarrolladores las herramientas necesarias para que las aplicaciones tengan buenas apariencias y la capacidad de sacar partido de todas las características de hardware de cada dispositivo.

Para ayudar a que el desarrollo sea eficiente, las herramientas de desarrollo que Android pone a disposición ofrecen un entorno de desarrollo Java con características avanzadas para el desarrollo, depuración y empaquetado de aplicaciones Android.



5.2.1 Características del entorno de desarrollo.

Las herramientas de desarrollo en Android (ADT) consisten en un añadido para el programa Eclipse que aporta un entorno de grado profesional para el desarrollo de aplicaciones. Es un entorno de programación totalmente Java con características que ayudan a crear, probar, depurar y empaquetar las aplicaciones Android creadas[14].

Cabe destacar las siguientes características:

- Entorno de desarrollo totalmente Java.
  - Uso específico para Android que integra la navegación entre recursos Java y XML.
  - Editor de XML para las fuentes XML de Android.
  - Herramientas de mejora, corrección y usabilidad.
  - Creado para soportar proyectos complejos.

- Generadores de interfaz de usuario gráfico.
  - Permite opciones de coger y situar elementos gráficos.
  - Visualización de interfaz de usuario en diferentes dispositivos. Varios temas.
  - Permite añadir elementos creados como componentes para la interfaz de usuario.

- Dispositivos móviles con opciones de desarrollo.
- Desarrollo en dispositivos físicos.
- Desarrollo en dispositivos virtuales.
  - Simulación de cualquier dispositivo o componente de hardware.

- Depurador de código muy potente.

6 MySQL - phpMyAdmin.

En este trabajo se ha elegido trabajar con un entorno web para la manipulación de la base de datos debido a que la aplicación creada no requiere características muy exigentes. Para ello se ha utilizado phpMyAdmin que es una herramienta de software escrita en PHP utilizada para la administración de MySQL desde la Web.

6.1 MySQL

Como anteriormente se explicó SQL es un lenguaje estándar internacional usado para crear y mantener bases de datos relacionales.

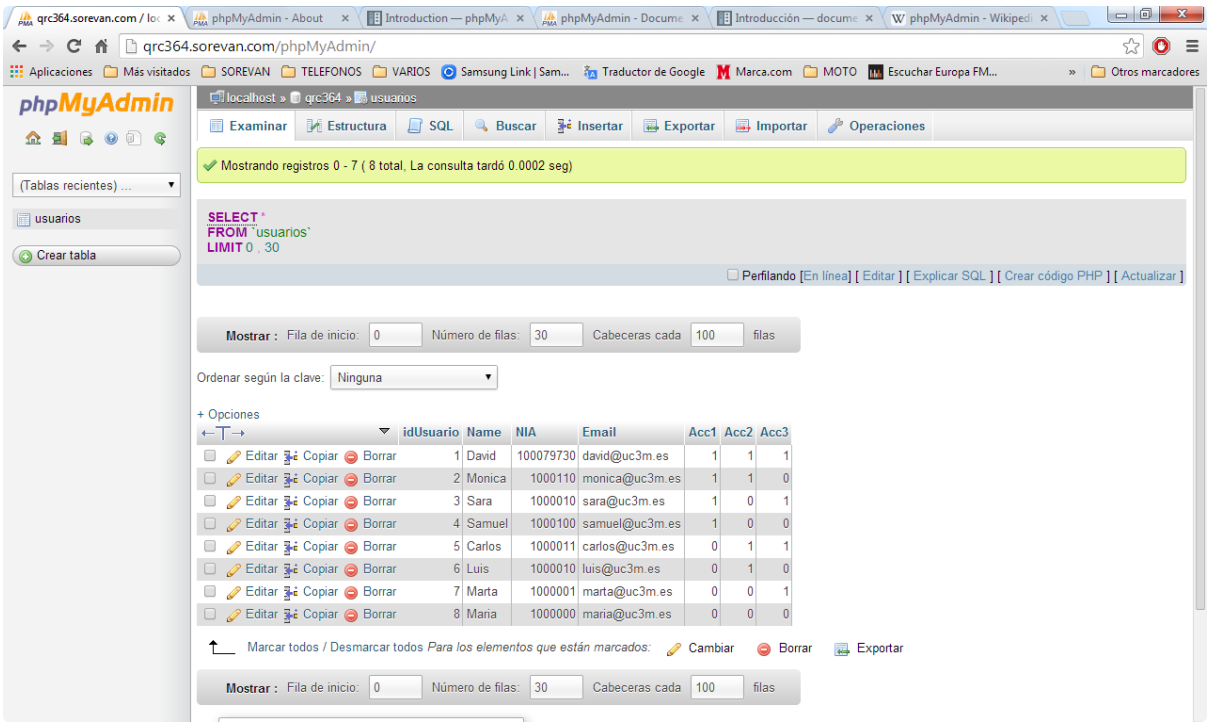
MySQL es un sistema de gestión de bases de datos multihilo, relacional, multiusuario y robusto. El servidor MySQL está diseñado para trabajar con entornos de producción críticos, con una carga de trabajo alta[15].

6.2 phpMyAdmin.

phpMyAdmin es una herramienta de administración de MySQL en la Web. Esta herramienta soporta un gran rango de operaciones en sistemas de manipulación de bases de datos (DBMSs) como MySQL, MariaDB y Drizzle. Las operaciones más comunes (como administrar bases de datos, tablas, columnas, relaciones, indexados, permisos, usuarios, etc) pueden ser llevadas a cabo a través de este interfaz[16].

phpMyAdmin es un projecto activo desde hace 15 años con un base de código estable y flexible.

El entorno de esta herramienta se muestra en la siguiente imagen:



6.2.1 Características phpMyAdmin.

Entre las características más destacables podemos encontrar:

- Interfaz web intuitiva.
- Capacidad para trabajar con la mayoría de características de MySQL.
- Posibilidad de importar archivos de datos de CSV y SQL.
- Posibilidad de exportar los datos en varios formatos: CSV, SQL, XML, PDF ISO/IEC 26300- OpenDocument Spreadsheet, Word, etc.
- Administración de múltiples servidores.
- Creación de gráficos en formato PDF del diseño de la base de datos.
- Creaciones de complejas consultas utilizando Query-By-example (QBE).
- Búsqueda global en la base de datos o en particular.
- Transformación de datos guardados a cualquier formato usando un conjunto de funciones predefinidas.
- Visualizar cambios en bases de datos, tablas y vistas.
- 62 idiomas disponibles.
- Etc.

7 Desarrollo de la aplicación.

Para realizar el control de accesos a un edificio se ha elegido realizarlo mediante la tecnología NFC, la utilización de elementos pasivos como los tags / etiquetas y el acceso mediante internet a una base de datos.

El sistema por lo tanto se compone de dos elementos físicos requeridos para el funcionamiento de la aplicación. La idea consiste en que cada usuario disponga de una etiqueta NFC o un Smartphone que tenga esta tecnología.

Como se mostró anteriormente los tag / etiquetas son capaces de guardar información en diferentes formatos. Para determinar los accesos permitidos de un usuario a una determinada área bastaría con leer el código contenido en esta etiqueta y obtener la información del usuario de una base de datos externa alojada en un servidor web.

Para poder leer el código de esta etiqueta y obtener la información de un usuario sería necesario el uso de un Smartphone con tecnología NFC.

En este caso también usaremos la aplicación para la escritura de las etiquetas, aunque en el caso práctico el sistema requeriría de permisos para tratar con estas etiquetas ya que de lo contrario cualquier usuario con esta aplicación podría asignarse todos los permisos de acceso al edificio.

A continuación se describirá cada una de las partes implicadas en el desarrollo de la aplicación Android que permite el uso y manipulación de las etiquetas NFC.

7.1 Permisos.

Lo primero que hay que indicar son los requisitos de los que debe disponer el dispositivo Android que va a alojar la aplicación. Por lo tanto los requisitos mínimos de hardware que debe tener el dispositivo para acceder a todas las capacidades de la aplicación son:

- Acceso a internet sin limites
- Controlar NFC

Ya que no todos los Smartphones Android del mercado disponen de la posibilidad de la tecnología NFC, esta aplicación con todas sus funciones no sería válida para cualquier dispositivo.

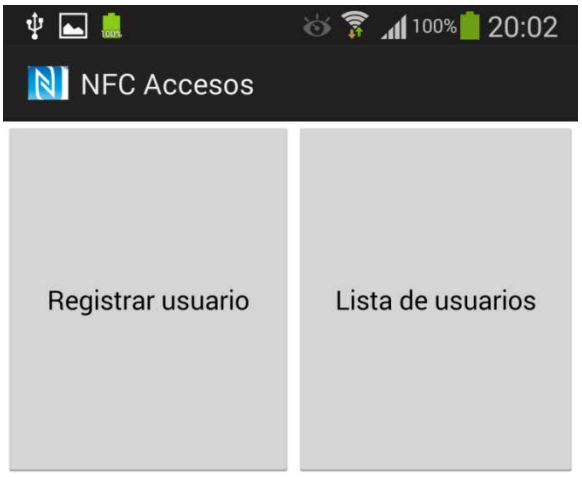
Además la aplicación requiere como mínimo una versión de software superior a 2.3 (Gingerbread) que incluye las bibliotecas necesarias para el uso de NFC.

7.2 Entorno de la aplicación.

La aplicación contiene cuatro diferentes pantallas, una que muestra las opciones disponibles, una para la manipulación de etiquetas y guardado de información en una base de datos, otra para la lectura de las etiquetas y una última como muestra de las entradas de la base de datos.

7.2.1 Pantalla de opciones.

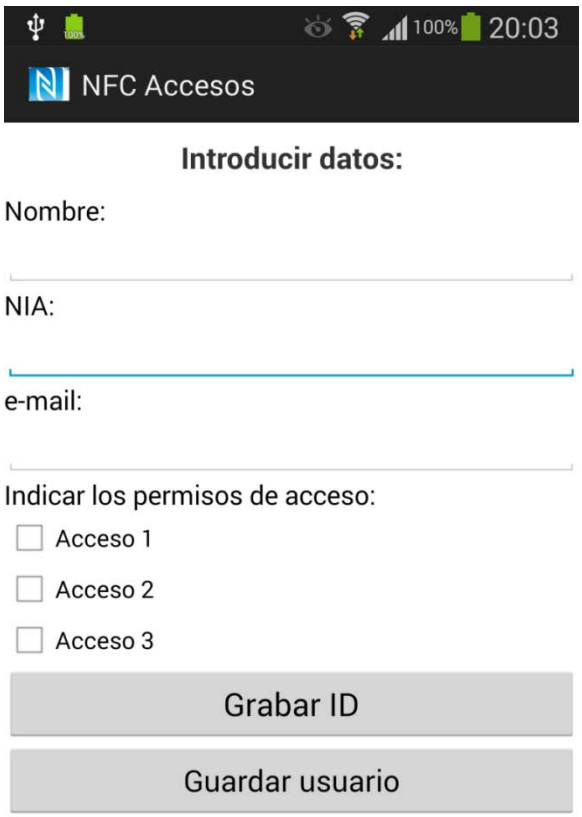
La pantalla de opciones se muestra en la siguiente imagen y en ella se muestran las opciones disponibles que permite la aplicación, esta pantalla es mostrada siempre y cuando se presione “Opciones” en el menú desplegable.



En esta pantalla se encuentra las dos opciones disponibles que se pueden realizar con la aplicación Android. Por una parte está el botón “Registrar usuario” que da acceso a la pantalla que permite el alta de un usuario nuevo y la grabación de datos en la etiqueta NFC y por otra parte está el botón que muestra la lista de usuarios dados de alta en la base de datos.

7.2.2 Pantalla de altas de usuario.

La siguiente imagen corresponde a la pantalla desde la que se pueden dar de alta los usuarios en la base de datos y también desde la que se pueden grabar las etiquetas, esta pantalla aparece siempre que se acceda a la aplicación desde el icono de la aplicación.



En la imagen se pueden apreciar todos los datos que contendrá la base de datos. Entre ellos se encuentran:

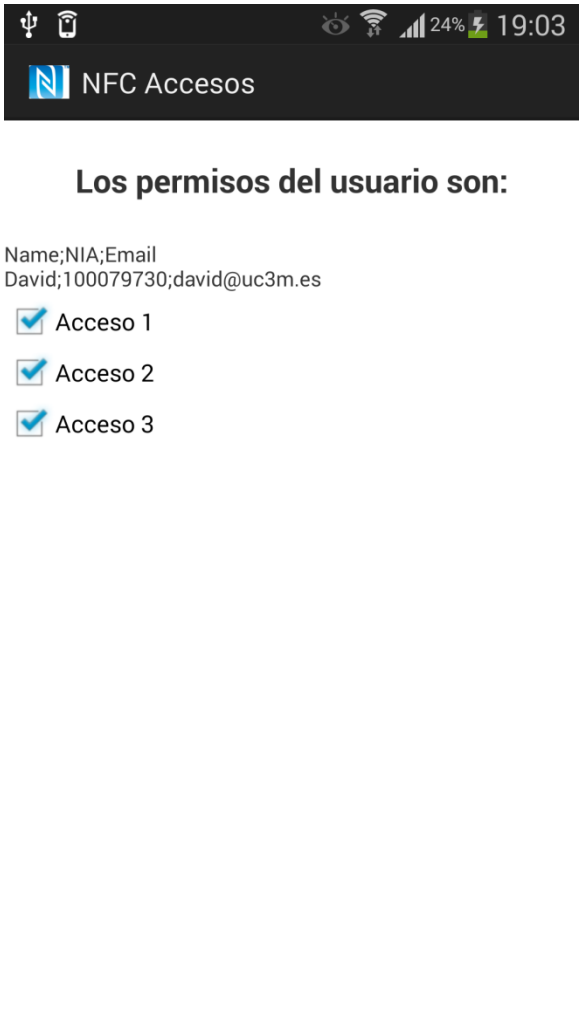
- Nombre: Nombre del usuario
- NIA: identificación personal y única de los usuarios.
- e-mail: Correo electrónico del usuario.
- Casillas check: utilizadas para indicar cada uno de los niveles de acceso del usuario.

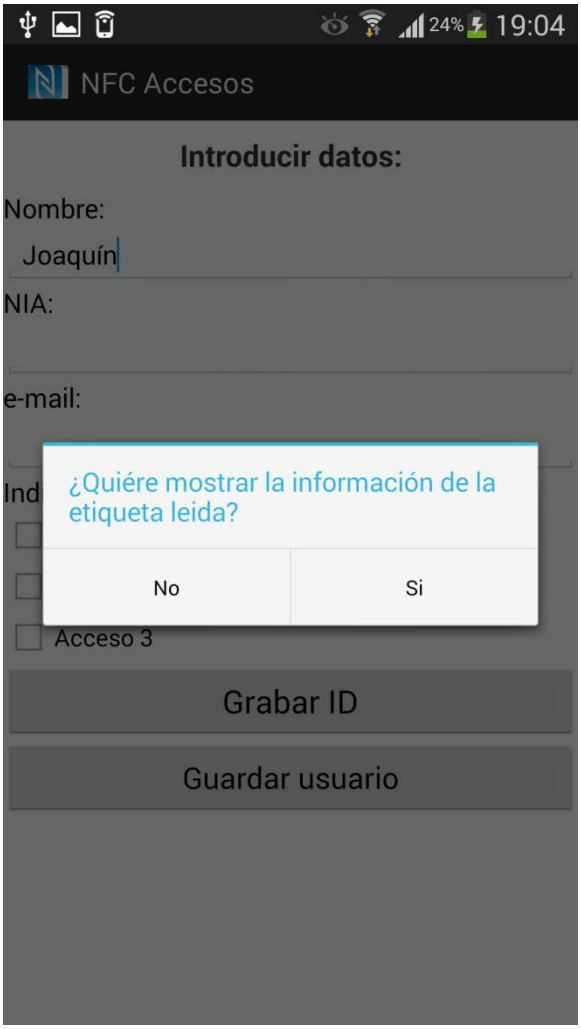
También podemos encontrar dos botones con los que seremos capaces de realizar las siguientes acciones:

- Grabar ID: este botón es el usado para grabar la etiqueta NFC con los datos del NIA. Se ha tomado el NIA debido a que ha sido considerado el único dato personal.
- Guardar usuario: en este caso la acción asignada a este botón es la de guardar todos los datos introducidos anteriormente en los diferentes campos en una base de datos externa almacenada en un servidor web.

7.2.3 Pantalla de información de tag o etiqueta de usuario.

Esta pantalla aparece siempre que se aproxime un tag o etiqueta al dispositivo móvil, mostrando así la información del usuario al que pertenece este tag identificador. También cabe la posibilidad de que durante la ejecución del programa (mostrándose cualquiera de las otras pantallas) se aproxime una de estas etiquetas, en cuyo caso la aplicación preguntara al usuario que acción queremos realizar. En las siguientes imágenes podemos ver el resultado de estas acciones.





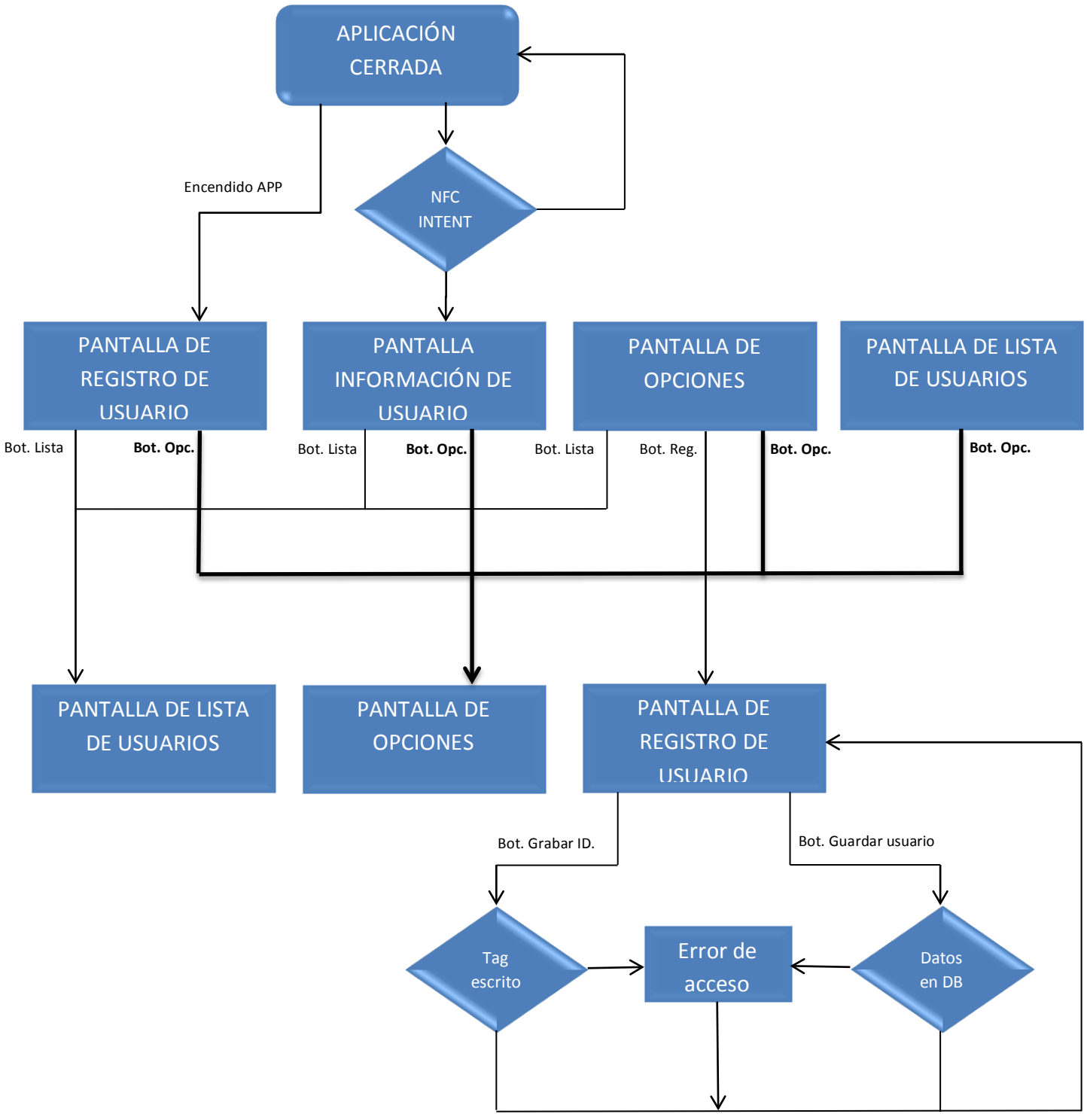
En la primera imagen aparece el resultado de acercar una etiqueta al dispositivo móvil. Esta etiqueta contiene grabado el NIA “100079730”, que es buscado por la aplicación en la base de datos y mostrado posteriormente en esta pantalla junto con los datos de este usuario.

En la segunda imagen aparece el cuadro de dialogo que se muestra cuando acercamos una etiqueta a nuestro Smartphone cuando tenemos abierta la aplicación. En este cuadro nos pregunta si queremos proseguir con la visualización de la etiqueta detectó el dispositivo o si por el contrario queremos no hacer nada con esta etiqueta detectada y proseguir con nuestra introducción de datos de usuario. Con este cuadro de dialogo nos cercioramos que no vamos a perder la información escrita hasta el momento.

7.2.4 Pantalla de lista de usuarios.

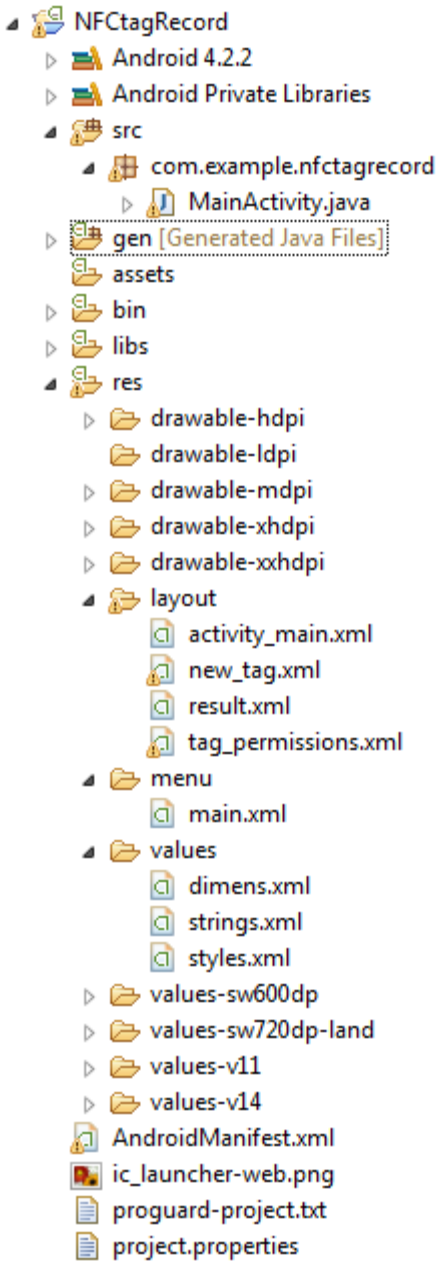
En esta pantalla se muestran los usuarios que han sido introducidos en el sistema. Una primera línea identifica el orden en el que aparecen estos datos. A continuación vemos una imagen de la pantalla.

7.3 Diagrama de bloques.



7.4 Explicación de la solución propuesta.

La aplicación Android desarrollada se compone de numerosos archivos, entre ellos se encuentran archivos de código, numerosos recursos, archivos en XML y en Java… A continuación se muestran los archivos más relevantes que forman parte de esta aplicación.



7.4.1 Manifest.

Para comenzar la explicación comenzaremos con el archivo "AndroidManifest.xml", este archivo contiene la información necesaria para que la aplicación Android pueda empezar. En este archivo se encarga de varias funciones entre las cuales se encuentran:

- En él se indican todos los componentes que funcionaran en esta aplicación.
- En este archivo también se indican todos los permisos de usuario que la aplicación requiere para su funcionamiento. Como por ejemplo el acceso a internet, lectura de la agenda telefónica, etc.
- En este archivo también se indica el nivel mínimo de API (versión de sistema operativo) que la aplicación requiere.
- También se declara el hardware y software requerido por la aplicación, como la cámara, bluetooth, NFC, etc.
- Librerías que no están por defecto entre las disponibles para Android.

A continuación se muestra el código correspondiente al archivo "AndroidManifest.xml" el cual tiene que estar en la raíz del directorio de la aplicación para su funcionamiento.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.nfctagrecord"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="9" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.NFC"/>


    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.nfctagrecord.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <!-- Se encarga de las etiquetas detectadas fuera de la aplicacion -->
            <intent-filter>
                <action android:name="android.nfc.action.NDEF_DISCOVERED" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="text/plain" />
```

```
            </intent-filter>
        </activity>

        <activity
        android:name=".Result" android:label="@string/tag_list">
        </activity>

    </application>

</manifest>
```

En este archivo encontramos que la versión mínima de API requerida para el funcionamiento de la aplicación es el API 10 la cual se conoce más comúnmente como 2.3 (Gingerbread), por lo tanto es necesario que para que esta aplicación funcione el sistema operativo del dispositivo tenga al menos esta versión. Esta limitación de software es debida a que no se incluyeron las bibliotecas correspondientes a NFC hasta la llegada de esta API. La versión mínima viene indicada de la siguiente forma:

```
android:minSdkVersion="10"
```

Si continuamos observando el código podemos ver los permisos que requiere la aplicación de hardware y software, en este caso la aplicación requiere del uso de Internet y del uso de la tecnología NFC, en el archivo "manifest" viene indicado del siguiente modo:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.NFC"/>
```

Lo siguiente que encontramos en este archivo es el apartado de "application" en el que se encuentran que se definen los recursos del icono de la aplicación, el nombre, la etiqueta, el tema, etc. También en este apartado encontramos la definición de las actividades que estarán incluidas en la aplicación.

La declaración de las actividades o "activity" sirven para indicar cada una de las partes de la interfaz visual del usuario. Si una actividad no está declarada en el "manifest" no podrá ser vista por el sistema y en consecuencia nunca podrá ser ejecutada.

En nuestro archivo "manifest" encontramos definidas las actividades principales de la aplicación y también encontramos dentro de las actividades los intentos o "intents". Un "intent" tiene asociado las capacidades del componente que define, por lo tanto, al definir un "intent" dentro de una actividad se hace posible que esta pueda responder al intento.

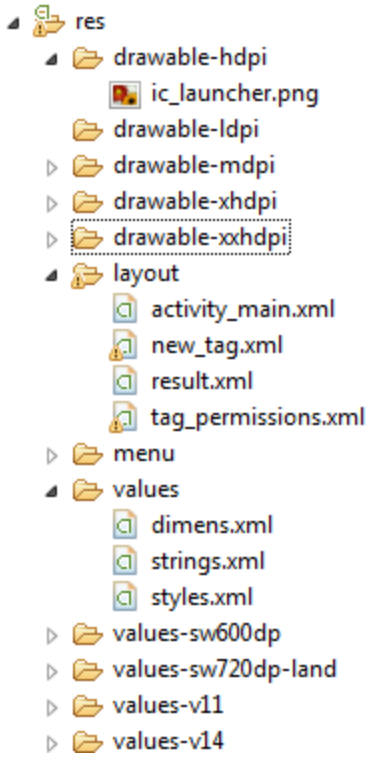
En el caso de esta aplicación hay que destacar la capacidad del uso de NFC cuyas señales son manejadas a través de "intents" tanto con la aplicación abierta como cerrada. El siguiente código muestra la declaración que hace posible este modo de empleo:

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
</intent-filter>
```

7.4.2 Archivos de recusos.

Toda clase de recursos que necesita la aplicación se pueden almacenar en la carpeta “res”, de este modo se externaliza el uso de todo recurso para el caso de necesidad de modificación o adaptación. En esta carpeta están los iconos de la aplicación, las cadenas de caracteres, estilos, el diseño o “layout” del entorno gráfico, recursos del menú de opciones, animaciones, etc.

En este programa se usan iconos, cadenas de caracteres, el menú de opciones, y diferentes “layouts” que se mostraron con anterioridad.



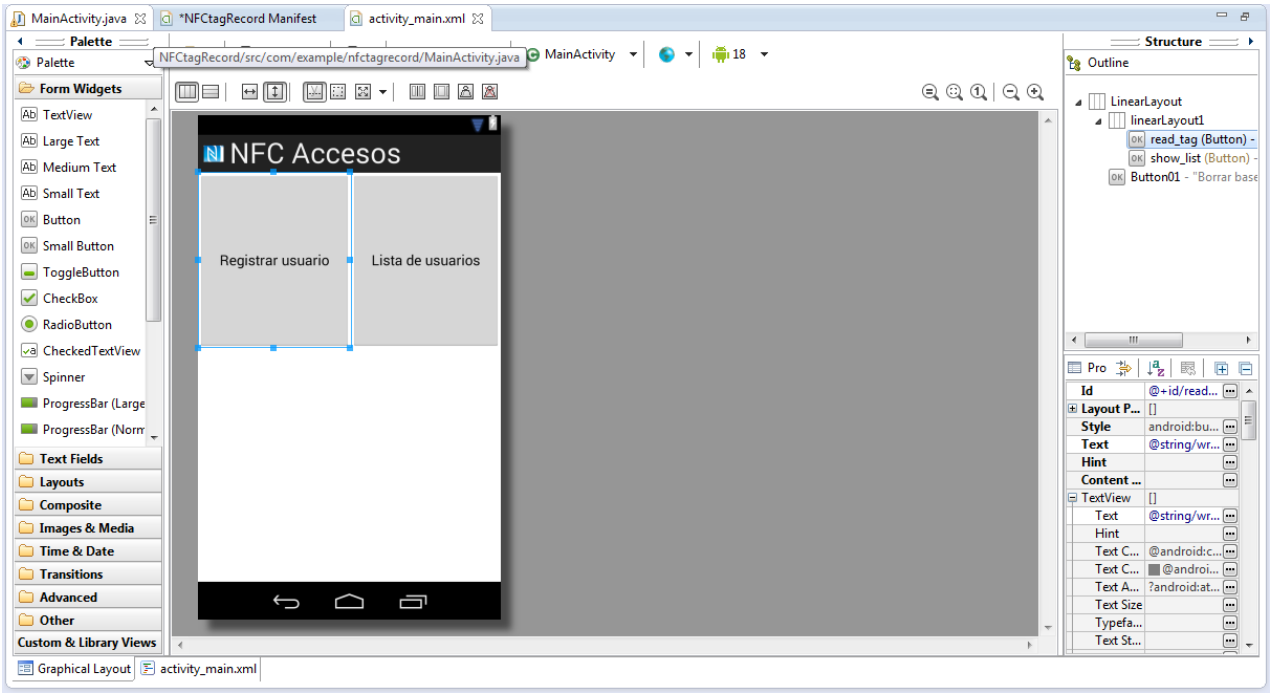
En la imagen superior se puede ver el contenido de la carpeta “res” donde se encuentran todos los recursos de la aplicación. A continuación se muestran algunos de ellos:

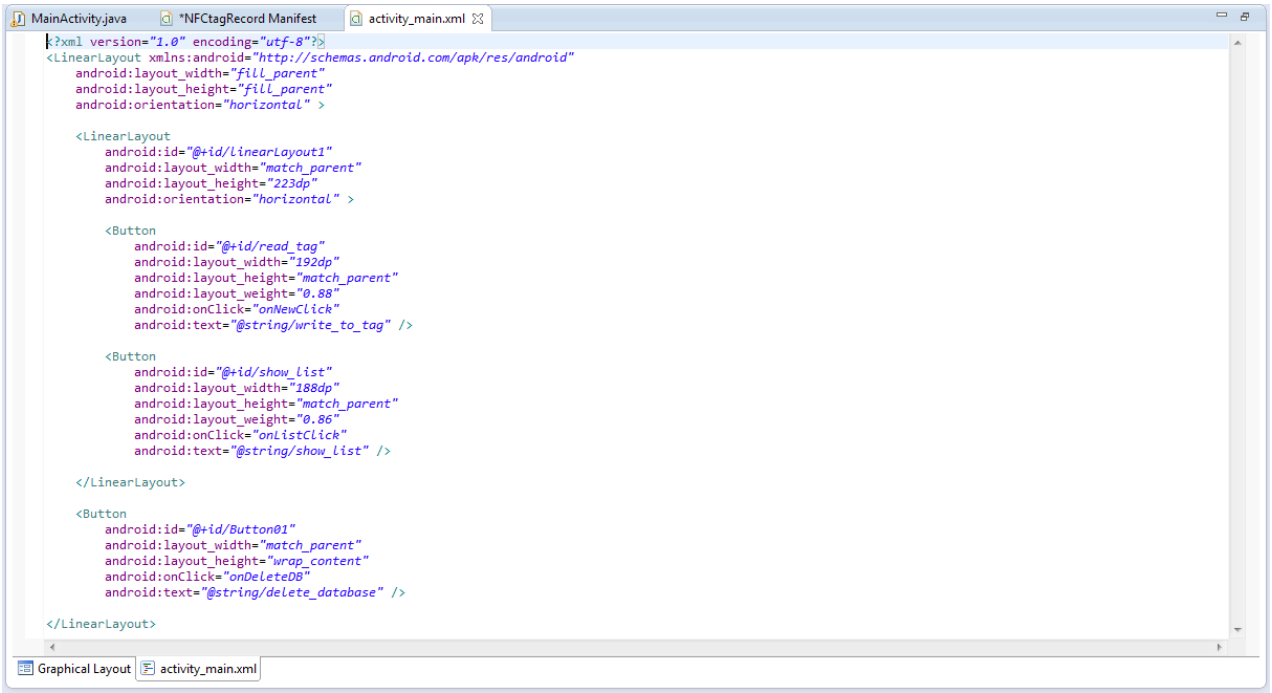
- Los iconos como por ejemplo “ic_launcher.png”

- Cadenas de caracteres en el archivo “strings.xml”. en este archivo se puede comprobar que es mucho más fácil modificar los datos, por ejemplo, para cambiar el idioma de la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">NFC Accesos</string>
    <string name="write_to_tag">Registrar usuario</string>
    <string name="read_tag">Read Tag</string>
    <string name="default_note_text">Edit me!</string>
    <string name="action_settings">Opciones</string>
    <string name="hello_world">Hello world!</string>
    <string name="tag_list">Lista de usuarios</string>
    <string name="show_list">Lista de usuarios</string>
    <string name="list"></string>
    <string name="close">Cerrar</string>
    <string name="listdb"></string>
    <string name="DatabaseHelper">DatabaseHelper</string>
    <string name="new_tag">Introducir datos:</string>
    <string name="name">Nombre:</string>
    <string name="nia">NIA:</string>
    <string name="save">Guardar usuario</string>
    <string name="default_name"></string>
    <string name="default_nia"></string>
    <string name="email">e-mail:</string>
    <string name="default_email"></string>
    <string name="permission">Indicar  los permisos de acceso:</string>
    <string name="acc1">Acceso 1</string>
    <string name="acc2">Acceso 2</string>
    <string name="acc3">Acceso 3</string>
    <string name="delete_database">Borrar base de datos</string>
    <string name="recordTag">Grabar ID</string>
    <string name="user_permissions">Los permisos del usuario son:</string>

</resources>
```

- Los archivos de “layout” que el entorno de eclipse nos permite visualizar y modificar de una forma muy visual o a través del archivo de código XML. Como ejemplo vemos el archivo activity_main.xml de ambas maneras.





7.4.3 Source o archivo de programa.

La base de la programación de la aplicación se encuentra en la carpeta "src". Esta carpeta contendrá el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc.

En el caso de esta aplicación se ha decidido crear un solo archivo en el que se encuentran todas las funciones que participan en la aplicación. Este archivo se llama "MainActivity.java" a continuación se describirá las funciones más importantes en el desarrollo del programa.

Lo primero indicar que para que este archivo comprenda todos los accesos a las diferentes bibliotecas hay que realizar unas importaciones de estas para su uso en el archivo. Como ejemplo se muestran las necesarias para el trabajo con NFC que son las más específicas que se encuentran en este código.

```
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.nfc.tech.NdefFormatable;
```

Después de las inclusiones de las bibliotecas la estructura de este archivo comenzaría con la definición de la clase que pertenece a la "activity" ya definida en el archivo "AndroidManifest.xml. Dentro de esta clase se incluirían variables y funciones que hacen funcionar el programa.

Las variables utilizadas en este programa son de diversos tipos, desde variables que sirven para conocer el estado de la aplicación, variables de cadenas de caracteres, variables visuales, variables de conexión o variables más específicas de Android como los "CheckBox".

7.4.3.1 OnCreate

En cuanto a las funciones que se encuentran en este archivo de código fuente encontramos la función "onCreate" que es empleada en todas las aplicaciones Android y se corresponde a la función que se ejecuta al comenzar la aplicación, tras pulsar el icono de abrir de la aplicación.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	mNfcAdapter = NfcAdapter.getDefaultAdapter(this);

	setContentView(R.layout.new_tag);

	mNIA = ((EditText) findViewById(R.id.editNIA));
	mNIA.addTextChangedListener(mTextWatcher);


	// Maneja todos los NFC intents recividos en esta actividad.
	mNfcPendingIntent = PendingIntent.getActivity(this, 0, new Intent
	(this,getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

	//Intent filters for reading a note from a tag or exchanging over p2p.
	IntentFilter ndefDetected = new IntentFilter(
		NfcAdapter.ACTION_NDEF_DISCOVERED);
	try {
		ndefDetected.addDataType("text/plain");
	} catch (MalformedMimeTypeException e) {
	}
	mNdefExchangeFilters = new IntentFilter[] { ndefDetected };

	// Intent filters for writing to a tag
	IntentFilter tagDetected = new IntentFilter(
		NfcAdapter.ACTION_TAG_DISCOVERED);
	mWriteTagFilters = new IntentFilter[] { tagDetected };

	// Inicialización de CheckBottons
	check1 = (CheckBox) findViewById(R.id.checkBox01);
	check1.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener(){
	public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){

		if (isChecked) {
			toast("Checkbox marcado!");
				acc1 = "1";
		} else {
			toast("Checkbox desmarcado!");
				acc1 = "0";
			}
		}
	});
	check2 = (CheckBox) findViewById(R.id.checkBox02);
	check2.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
	public void onCheckedChanged(CompoundButton buttonView,boolean isChecked) {

		if (isChecked) {
```

```
                toast("Checkbox marcado!");
                        acc2 = "1";
            } else {
                toast("Checkbox desmarcado!");
                        acc2 = "0";
                }
            }
        });
        check3 = (CheckBox) findViewById(R.id.checkBox03);
        check3.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,boolean isChecked) {

            if (isChecked) {
                toast("Checkbox marcado!");
                acc3 = "1";
            } else {
                toast("Checkbox desmarcado!");
                acc3 = "0";
                }
            }
        });

}
```

En esta función se inicia la pantalla que aparecerá en la aplicación en primer lugar, que en este caso es la pantalla de altas de usuario.

```
setContentView(R.layout.new_tag);
```

Se inicializan también los componentes visuales de la aplicación que van a interactuar con la aplicación, como pueden ser los campos de texto o los cuadros de CheckBox.

También en esta función se definen los tipos de "intent" encargados del trabajo con la tecnología NFC, que pueden ser para la lectura de información contenida en etiquetas o para la escritura de información en las etiquetas.

En esta función se inicializan las variables "CheckBox" y se controla su estado, de modo que al ser pulsada la casilla se guarda el dato de activo o no activo en una variable de estado de acceso, en este caso se han empleado variables de tipo cadena de caracteres (String acc1) en las que se guarda la información de un "1" en caso de tener permitido el acceso a esta área o un "0" en caso de no tener acceso.

```
check1 = (CheckBox) findViewById(R.id.checkBox01);
check1.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

	if (isChecked) {
		toast("Checkbox marcado!");
		acc1 = "1";
	} else {
		toast("Checkbox desmarcado!");
		acc1 = "0";
		}
	}
});
```

7.4.3.2 onResume

La siguiente función básica que podemos encontrar en este archivo de código fuente, es la función "onResume". Esta función sirve para abrir la aplicación de un modo especial, ya que sólo es posible abrirla de este modo en caso de que el dispositivo reciba un "intent" de información de NFC, es decir que sólo funcionará de este modo en caso de que acerquemos una etiqueta al dispositivo cuando está cerrada la aplicación. El código de esta aplicación se muestra a continuación.

```
@Override
protected void onResume() {
	super.onResume();
	mResumed = true;
	// NFCtagrecord received from Android
	if(NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {

		byte[] payload = messages[0].getRecords()[0].getPayload();
		setNoteBody(new String(payload));

		String bNIA = new String(messages[0].getRecords()[0].getPayload());

		setContentView(R.layout.tag_permissions);

		String buscar="select Name, NIA, email from usuarios where
		NIA='"+bNIA+"';";
		cargarConfiguracion();
		SQLEjecutar =buscar;
		conectarBDMySQL(usuarioMySQL, contrasenaMySQL, ipServidorMySQL,
		puertoMySQL, catalogoMySQL);
		String resultadoSQL = ejecutarConsultaSQL(false, buscar);
```

```
            TextView textResultadoSQL = (TextView)
            findViewById(R.id.tag_permission_user);
            textResultadoSQL.setText(resultadoSQL);


            String comprobar1 = "select Acc1 from usuarios where
            NIA='"+bNIA+"';";
            String comprobar2 = "select Acc2 from usuarios where
            NIA='"+bNIA+"';";
            String comprobar3 = "select Acc3 from usuarios where
            NIA='"+bNIA+"';";

            SQLEjecutar = comprobar1;
            String Acc1result= ejecutarConsultaSQL(false, comprobar1);
            check1tag = (CheckBox) findViewById(R.id.checkBox1tag);

            SQLEjecutar = comprobar2;
            String Acc2result= ejecutarConsultaSQL(false, comprobar2);
            check2tag = (CheckBox) findViewById(R.id.checkBox2tag);

            SQLEjecutar = comprobar3;
            String Acc3result= ejecutarConsultaSQL(false, comprobar3);
            check3tag = (CheckBox) findViewById(R.id.checkBox3tag);

            int aux1 = lectorString (Acc1result, 2);
            int aux2 = lectorString (Acc2result, 1);
            int aux3 = lectorString (Acc3result, 1);

            check1tag.setChecked(check(aux1));
            check2tag.setChecked(check(aux2));
            check3tag.setChecked(check(aux3));

            setIntent(new Intent()); // Consume this intent.
      }
      enableNdefExchangeMode();
}
```

En esta función se obtienen los datos recibidos de la etiqueta NFC y se realiza una búsqueda en la base de datos externa para obtener los datos asociados a la etiqueta del usuario detectada. La condición que hace posible la activación del programa a través de la comunicación NFC es la siguiente:

```
if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {
            NdefMessage[] messages = getNdefMessages(getIntent());
            byte[] payload = messages[0].getRecords()[0].getPayload();
            setNoteBody(new String(payload));

            String bNIA = new String(messages[0].getRecords()[0].getPayload());
```

La obtención de datos sólo es posible si se ha detectado un mensaje de una tarjeta NFC, a partir de ese momento se trata este mensaje y se guarda el contenido en una variable de cadena de caracteres. Es esta cadena de caracteres la que se utiliza para realizar la posterior búsqueda en la base de datos del servidor. Esta búsqueda es posible gracias a la conexión a Internet de la aplicación. A continuación se muestra como se trabajaría con el servidor:

```
String buscar="select Name, NIA, email from usuarios where NIA='"+bNIA+"';";
cargarConfiguracion();
SQLEjecutar =buscar;
conectarBDMySQL(usuarioMySQL,  contrasenaMySQL,  ipServidorMySQL,  puertoMySQL,
catalogoMySQL);
String resultadoSQL = ejecutarConsultaSQL(false, buscar);
TextView textResultadoSQL = (TextView) findViewById(R.id.tag_permission_user);
textResultadoSQL.setText(resultadoSQL);
```

En estas líneas de código vemos la intervención de varias funciones que sirven para tratar con la base de datos. En este caso se hace una consulta en lenguaje SQL para obtener unos datos concretos de nombre, NIA e email de un usuario con un NIA = “bNIA”. Estos datos obtenidos son posteriormente mostrados por la aplicación.

También se realizan consultas sucesivas para obtener los permisos de acceso del usuario buscado.

7.4.3.3 Trabajo con NFC

En las siguientes líneas se tratara cómo es posible la coordinación de la comunicación NFC en la aplicación.

Lo primero hay que indicar que hay dos posibles modos de interactuación con un tag o etiqueta NFC, en modo escritura y en modo lectura. En la aplicación es posible conocer el modo de interactuación con la etiqueta NFC gracias a una variable de estado de tipo booleana (mWriteMode) que en caso de tener un resultado verdadero significara que la interacción que habrá con la etiqueta será para escritura.

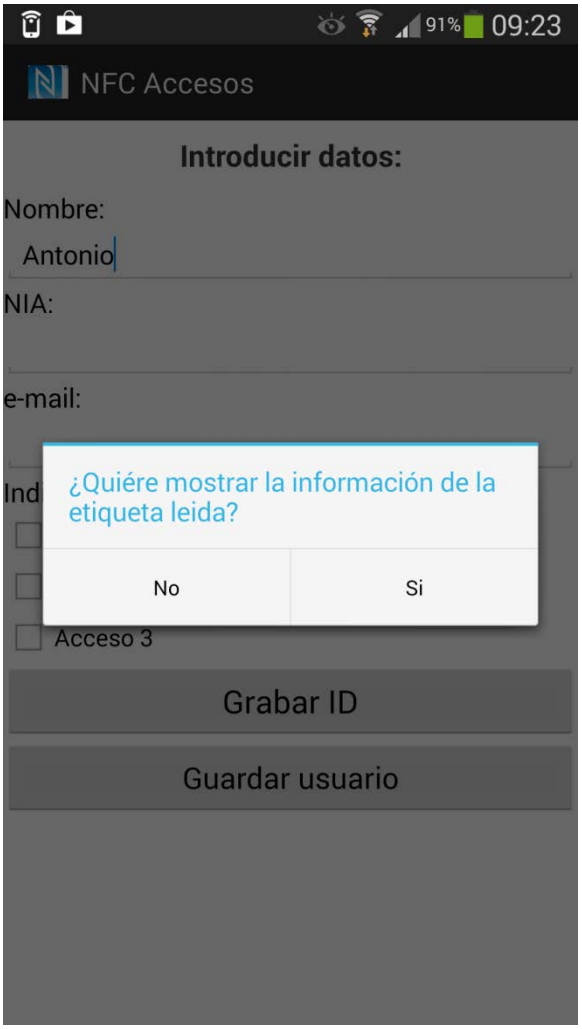
Para hacer posible que la aplicación sea capaz de reconocer las comunicaciones con los tags NFC se han definido dos tipos de “intents”. Uno que es válido para la escritura y otro para la lectura. De este modo la aplicación sabe interpretar que función tiene que hacer para cada “intent”. A continuación se muestra la declaración de estos:

```
IntentFilter[] mWriteTagFilters;
IntentFilter[] mNdefExchangeFilters;
```

Teniendo en cuenta el estado de la variable booleana mWriteMode y los "intent" de NFC recibidos la aplicación interpreta esta información para poder establecer el modo de uso. La función siguiente se encarga del tratamiento de estas variables y activa el funcionamiento de otras funciones relativas al trabajo con estas etiquetas.

```
@Override
protected void onNewIntent(Intent intent) {
	// Modo lectura NDEF
	if(!mWriteMode &&
		NfcAdapter.ACTION_NDEF_DISCOVERED.equals(intent.getAction())) {
			NdefMessage[] msgs = getNdefMessages(intent);
			promptForContent(msgs[0]);

		}

	// Modo de escritura
	if (mWriteMode &&
		NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
		Tag detectedTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
		writeTag(getNoteAsNdef(), detectedTag);

	}
}
```

Cuando es recibido un “intent” y la variable mWriteMode tiene como valor “falso”, accederíamos al modo de lectura en la que encontramos la función “promptForConten()” la cual se encarga de mostrar la información de la etiqueta, aunque siempre preguntando si es efectivamente lo que desea el usuario que maneja la aplicación, ya que dejaría de ver la pantalla actual en la que está la aplicación. En la siguiente imagen podemos ver la pregunta efectuada por la aplicación:



Para que todo esto sea posible se muestra el código de la función "promptForContent()" que incluye la pregunta que efectúa la aplicación, dos botones para poder responder a esta pregunta, y las funciones necesarias para mostrar la información de la etiqueta, como vimos anteriormente en la función "onResume".

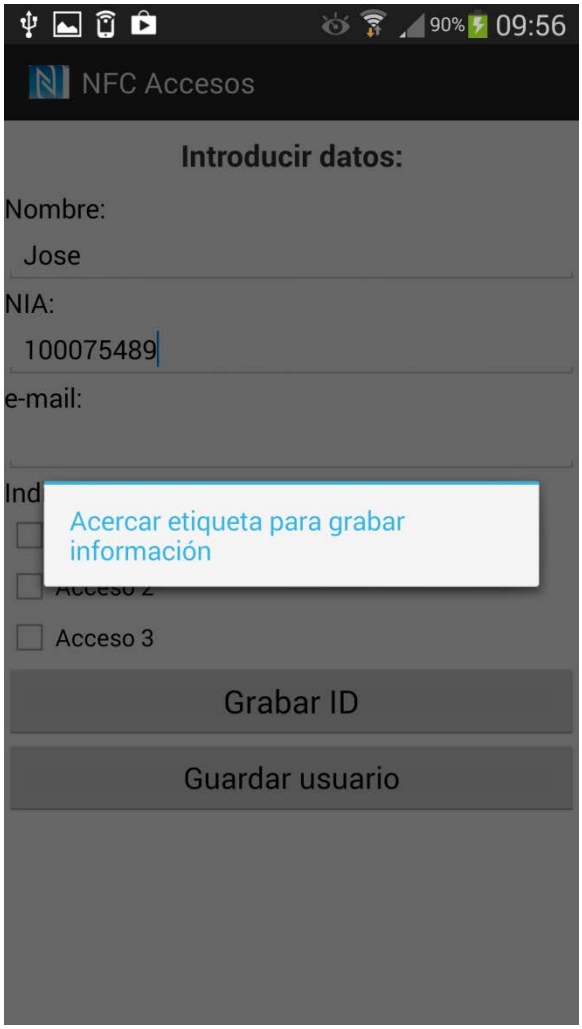
```
private void promptForContent(final NdefMessage msg) {
        new AlertDialog.Builder(this)
                .setTitle("¿Quiére mostrar la información de la etiqueta leida?")
                .setPositiveButton("Si", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface arg0, int arg1) {
                        String body = new String(msg.getRecords()[0]
                                .getPayload());
                        setNoteBody(body);
                        tag = body;

                        String bNIA = body;

                        setContentView(R.layout.tag@permissions);

                        String buscar = "select Name, NIA, email from usuarios
                                 where NIA='"+ bNIA + "';";
                        cargarConfiguracion();
                        SQLEjecutar = buscar;
                        conectarBDMySQL(usuarioMySQL, contrasenaMySQL,
                                ipServidorMySQL, puertoMySQL, catalogoMySQL);
                        String resultadoSQL = ejecutarConsultaSQL(false, buscar);
                        TextView textResultadoSQL =
                        (TextView) findViewById(R.id.tag@permission@user);
                        textResultadoSQL.setText(resultadoSQL);

                        String comprobar1 ="select Acc1 from usuarios where NIA='"
                                + bNIA + "';";
                        String comprobar2 ="select Acc2 from usuarios where NIA='"
                                + bNIA + "';";
                        String comprobar3 ="select Acc3 from usuarios where NIA='"
                                + bNIA + "';";

                        SQLEjecutar = comprobar1;
                        String Acc1result = ejecutarConsultaSQL(false,
                                comprobar1);
                        check1tag = (CheckBox) findViewById(R.id.checkBox7tag);

                        SQLEjecutar = comprobar2;
                        String Acc2result = ejecutarConsultaSQL(false,
                                comprobar2);
                        check2tag = (CheckBox) findViewById(R.id.checkBox8tag);

                        SQLEjecutar = comprobar3;
                        String Acc3result = ejecutarConsultaSQL(false,
                                comprobar3);
                        check3tag = (CheckBox) findViewById(R.id.checkBox9tag);
```

```
                    int aux1 = lectorString(Acc1result, 2);
                    int aux2 = lectorString(Acc2result, 1);
                    int aux3 = lectorString(Acc3result, 1);

                    check1tag.setChecked(check(aux1));
                    check2tag.setChecked(check(aux2));
                    check3tag.setChecked(check(aux3));

                    setIntent(new Intent()); // Consume this intent.

                }
            })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface arg0, int arg1) {

                }
            }).show();
}
```

Una vez mostrado como se realiza la lectura de etiquetas NFC a continuación se indicara cómo funciona el proceso de escritura de estas etiquetas. Para que esto suceda es necesario que se reúnan dos condiciones, la primera que la variable “mWriteMode” tenga un valor verdadero, y que se reciba un “intent”. Para que la aplicación pueda grabar un tag es necesario pulsar el botón “Grabar ID” de la pantalla de altas de usuario como ya se indicó anteriormente. Al pulsar este botón se activa el modo de escritura y también el tipo de “intent” que pueda recibir la aplicación. En las siguientes líneas de código se muestra la función que es activada cuando es pulsado el botón “Grabar ID”:

```
public void onRecordTag(View botton) {
      // Write to a tag for as long as the dialog is shown.
      disableNdefExchangeMode();
      enableTagWriteMode();

      new AlertDialog.Builder(MainActivity.this)
                  .setTitle("Acercar etiqueta para grabar información")
                  .setOnCancelListener(new
                              DialogInterface.OnCancelListener() {
                        @Override
                        public void onCancel(DialogInterface dialog) {
                              disableTagWriteMode();
                              enableNdefExchangeMode();
                        }
                  }).create().show();
}
```

En estas líneas se aprecia primeramente las funciones que modifican la variable "mWriteMode" con la función "enableTagWriteMode()" y también encontramos la función encargada de cambiar el tipo de "intent" recibido que en este caso sería "disableNdefExchangeMode". Después de estos dos cambios en el estado del programa cualquier etiqueta que sea acercada al dispositivo será interpretada para ser escrita. Lo siguiente que encontramos en esta función es el interfaz que indica al usuario de la aplicación que acerque una etiqueta para ser escrita. En caso de pulsar el botón de cancelar del dispositivo quedaría anulado los cambios realizados en el estado de la aplicación y volvería a estar activo el modo de lectura. A continuación se muestra como sería la pantalla que indicaría realizar la acción de acercar el tag.



No obstante en esta función no se realiza el grabado de la información en la etiqueta, para ello entran en juego otras dos funciones. La primera función es la que ha sido explicada anteriormente, "onNewIntent()", que activa la función que escribe la etiqueta. La función encargada de escribir el tag es "writeTag" y además de escribir las etiquetas realiza numerosas comprobaciones para indicar al usuario de la aplicación si la etiqueta que está leyendo es válida o no para la escritura, o también de si fue posible o no la escritura. Las siguientes líneas de código muestran esta función encargada de grabar la información en las etiquetas:

```
boolean writeTag(NdefMessage message, Tag tag) {
	int size = message.toByteArray().length;

	try {
		Ndef ndef = Ndef.get(tag);
		if (ndef != null) {
			ndef.connect();

			if (!ndef.isWritable()) {
				toast("Tag de sólo lectura.");
				return false;
			}
			if (ndef.getMaxSize() < size) {
				toast("La capacidad del tag es " + ndef.getMaxSize()
					+ " bytes, el mensaje es  " + size + " bytes.");
				return false;
			}
			ndef.writeNdefMessage(message);
			toast("Mensaje escrito satisfactoriamente.");
			return true;
		} else {
			NdefFormatable format = NdefFormatable.get(tag);
			if (format != null) {
				try {
					format.connect();
					format.format(message);
					toast("Tag formateada y mensaje escrito");
					return true;
				} catch (IOException e) {
					toast("Error en el formateo.");
					return false;
				}
			} else {
				toast("La etiqueta no soporta NDEF.");
				return false;
			}
		}
	} catch (Exception e) {
		toast("Error en la escritura de la etiqueta");
	}

	return false;
}
```

7.4.3.4 Trabajo con base de datos externa.

Para que la aplicación sea capaz de acceder a una base de datos externa son necesarios una serie de requisitos:

- Un dispositivo que tenga conexión a Internet.
- La aplicación debe tener permisos para aprovechar la conexión a Internet del dispositivo.
- Un servidor disponible para albergar la base de datos.
- Unos permisos de acceso para la base de datos.

En esta aplicación se aúnan todos los requisitos indicados, por lo tanto a continuación se mostrará cómo es posible esta conexión con el servidor externo. Para explicar el funcionamiento se tomará como ejemplo la función encargada de guardar datos en la base de datos.

Para guardar la información de un usuario en la base de datos es necesario pulsar el botón "Guardar usuario" como se explicó anteriormente. Al pulsar este botón se ejecuta la función encargada de tomar los datos introducidos y posteriormente enviarlos a la función encargada de guardarlos en la base de datos. La función activada al pulsar el botón es la siguiente:

```
public void onSave(View botton) {

	TextView sname = (TextView) findViewById(R.id.editName);
	TextView snia = (TextView) findViewById(R.id.editNIA);
	TextView semail = (TextView) findViewById(R.id.editemail);
	insertTag(sname.getText().toString(), snia.getText().toString(), acc1,
		acc2, acc3, semail.getText().toString());

}
```

Por lo tanto la función "onSave()" activada con el botón acabaría enviando a la aplicación a la función "insertTag()" encargada de la escritura en la base de datos. En esta función entran en juego todas las funciones necesarias para poder conectar con la base de datos. Las siguientes líneas de código se corresponden a la función "insertTag()".

```
public void insertTag(String name, String tag, String acc1, String acc2,
            String acc3, String email) {

      String insert = "insert into usuarios values(NULL, '"
            + name
            + "','"
            + tag
            + "', '"
            + email
            + "', '"
            + acc1
            + "', '"
            + acc2
            + "', '"
            + acc3 +"')";
      cargarConfiguracion();
      SQLEjecutar = insert;
      conectarBDMySQL(usuarioMySQL, contrasenaMySQL, ipServidorMySQL,
            puertoMySQL, catalogoMySQL);
      ejecutarConsultaSQL(true, insert);

}
```

En esta función se aprecian las tres funciones necesarias para la conexión con la base de datos, estas funciones realizan las siguientes acciones:

- "cargarConfiguracion()" -> esta función es la encargada de cargar los datos necesarios para que sea posible la conexión con el servidor. Estos datos son:
  - Nombre del servidor o IP
  - Contraseña de acceso
  - Puerto de acceso
  - Usuario
  - Catálogo

- "conectarBDMySql()" -> esta función establece el enlace con la base de datos.
- "ejecutarConsutaSQL" -> por ultimo esta función envía la consulta al servidor, el cual trabaja sobre la consulta realizada.

A continuación se mostraran estas tres funciones involucradas en la conexión con el servidor:

```
public void cargarConfiguracion() {
	catalogoMySQL = "****";
	ipServidorMySQL = "****";
	contrasenaMySQL = "****";
	puertoMySQL = "****";
	usuarioMySQL = "****";
}
```

Los datos de conexión son restringidos.

```
public void conectarBDMySQL(String usuario, String contrasena, String ip,
		String puerto, String catalogo) {
	if (usuario == "" || puerto == "" || ip == "") {
		AlertDialog.Builder alertDialog = new AlertDialog.Builder(
				MainActivity.this);
		alertDialog.setMessage("Antes de establecer la conexiÃ³n "
				+ "con el servidor "
				+ "MySQL debe indicar los datos de conexiÃ³n "
				+ "(IP, puerto, usuario y contraseÃ±a).");
		alertDialog.setTitle("Datos conexiÃ³n MySQL");
		alertDialog.setIcon(android.R.drawable.ic_dialog_alert);
		alertDialog.setCancelable(false);
		alertDialog.setPositiveButton("Aceptar",
				new DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog, int
						which) {
						}
				});
		alertDialog.show();
	} else {
		String urlConexionMySQL = "";
		if (catalogo != "")
			urlConexionMySQL = "jdbc:mysql://" + ip + ":" + puerto + "/"
					+ catalogo;
		else
			urlConexionMySQL = "jdbc:mysql://" + ip + ":" + puerto;
		if (usuario != "" & contrasena != "" & ip != "" & puerto != "") {
			try {
				Class.forName("com.mysql.jdbc.Driver");
				conexionMySQL = DriverManager.getConnection(
						urlConexionMySQL, usuario, contrasena);
			} catch (ClassNotFoundException e) {
				Toast.makeText(getApplicationContext(),
						"Error: " + e.getMessage(),
							Toast.LENGTH_SHORT)
						.show();
			} catch (SQLException e) {
				Toast.makeText(getApplicationContext(),
						"Error: " + e.getMessage(),
							Toast.LENGTH_SHORT)
						.show();
			}
```

```
            }
        }
```

En la función de "conectarDBMySQL()" encontramos las comprobaciones que son realizadas para establecer la conexión, de modo que si hubiera algún tipo de error a la hora de realizar esta comunicación aparecería un mensaje de error.

Por último se muestra la función que realiza la consulta en la base de datos, que como peculiaridad tiene un parámetro el cual sirve para indicar a la función si se va a realizar una consulta en la base de datos o si por el contrario lo que se va a realizar es una modificación.

```
public static String ejecutarConsultaSQL(Boolean SQLModificacion, String context)
    {
        try
             {
          String resultadoSQL = "";
          //ejecutamos consulta SQL de seleccIÃ³n (devuelve datos)
         if (!SQLModificacion)
         {
                Statement st = conexionMySQL.createStatement();
                ResultSet rs = st.executeQuery(SQLEjecutar);

                      Integer numColumnas = 0;

                //nÃºmero de columnas (campos) de la consula SQL
                numColumnas = rs.getMetaData().getColumnCount();

                //obtenemos el tÃ-tulo de las columnas
                for (int i = 1; i <= numColumnas; i++)
                {
                      if (resultadoSQL != "")
                            if (i < numColumnas)
                                  resultadoSQL = resultadoSQL
                                +      s.getMetaData().getColumnName(i).toString()
                                +";";
                            else
                                  resultadoSQL = resultadoSQL
                                + rs.getMetaData().getColumnName(i).toString();
                      else
                            if (i < numColumnas)
                                  resultadoSQL =
                                rs.getMetaData().getColumnName(i).toString()
                                + ";";
                            else
                                  resultadoSQL =
                                  rs.getMetaData().getColumnName(i).toString();
                }


                //mostramos el resultado de la consulta SQL
                while (rs.next())
                {
                      resultadoSQL = resultadoSQL + "\n";

                      //obtenemos los datos de cada columna
```

```
                    for (int i = 1; i <= numColumnas; i++)
                {
                    if (rs.getObject(i) != null)
                    {
                        if (resultadoSQL != "")
                            if (i < numColumnas)
                                resultadoSQL = resultadoSQL +
                                        rs.getObject(i).toString()
                                        +";";
                            else
                                resultadoSQL = resultadoSQL +
                                        rs.getObject(i).toString();
                        else
                            if (i < numColumnas)
                                resultadoSQL = rs.getObject(i).toString()
                                    + ";";
                            else
                                resultadoSQL =
                                    rs.getObject(i).toString();
                    }
                    else
                    {
                        if (resultadoSQL != "")
                            resultadoSQL = resultadoSQL + "null;";
                        else
                            resultadoSQL = "null;";
                    }
                }
                resultadoSQL = resultadoSQL + "\n";
            }
            st.close();
            rs.close();
            }
        // consulta SQL de modificaciÃ³n de
        // datos (CREATE, DROP, INSERT, UPDATE)
        else
        {
            int numAfectados = 0;
            Statement st = conexionMySQL.createStatement();
            numAfectados = st.executeUpdate(SQLEjecutar);
            resultadoSQL = "Registros afectados: "
                + String.valueOf(numAfectados);
            st.close();
        }
        return resultadoSQL;
            }

        catch (Exception e)
        {
            return "";
        }
}
```

8 Resultado.

En este apartado se incluyen las pruebas realizadas sobre la aplicación y el entorno de la base de datos una vez terminada la aplicación.

El objetivo de estas verificaciones finales es la comprobación de que la manipulación de la base de datos es correcta y de que la comunicación que se establece entre el Smartphone y las etiquetas es el correcto.

Todas las pruebas efectuadas, tanto en el desarrollo de la aplicación como las realizadas una vez concluida han sido realizadas en un dispositivo con el sistema Android. En este caso ha sido probada la aplicación con un Samsung Galaxy S4 GT-I9505 con dos versiones de software distintas debido a una actualización, primeramente con la versión 4.2.2 y después con la versión más actual del dispositivo, la 4.3.

8.1 Objetivos de las pruebas.

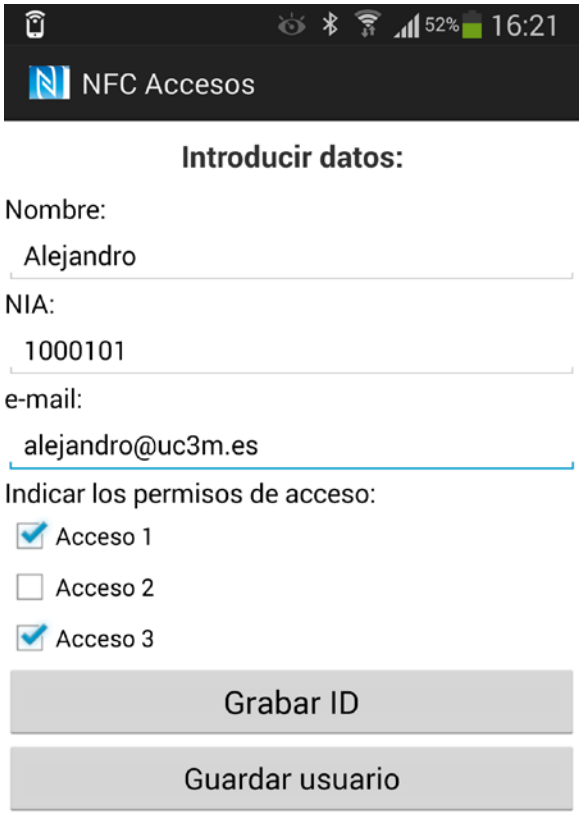
Para poder comprobar el correcto funcionamiento de la aplicación, las etiquetas NFC y la manipulación de la base de datos con phpMyAdmin se han realizado las siguientes pruebas y simulaciones:

- Alta de un nuevo usuario en el sistema a través de la aplicación Android.
- Modificación de la información de un usuario con phpMyAdmin.
- Grabación de un tag con el Smartphone.
- Lectura de un tag con el Smartphone con la aplicación en ejecución y en estado de reposo.
- Comprobación de contenido de la base de datos desde la aplicación Android.

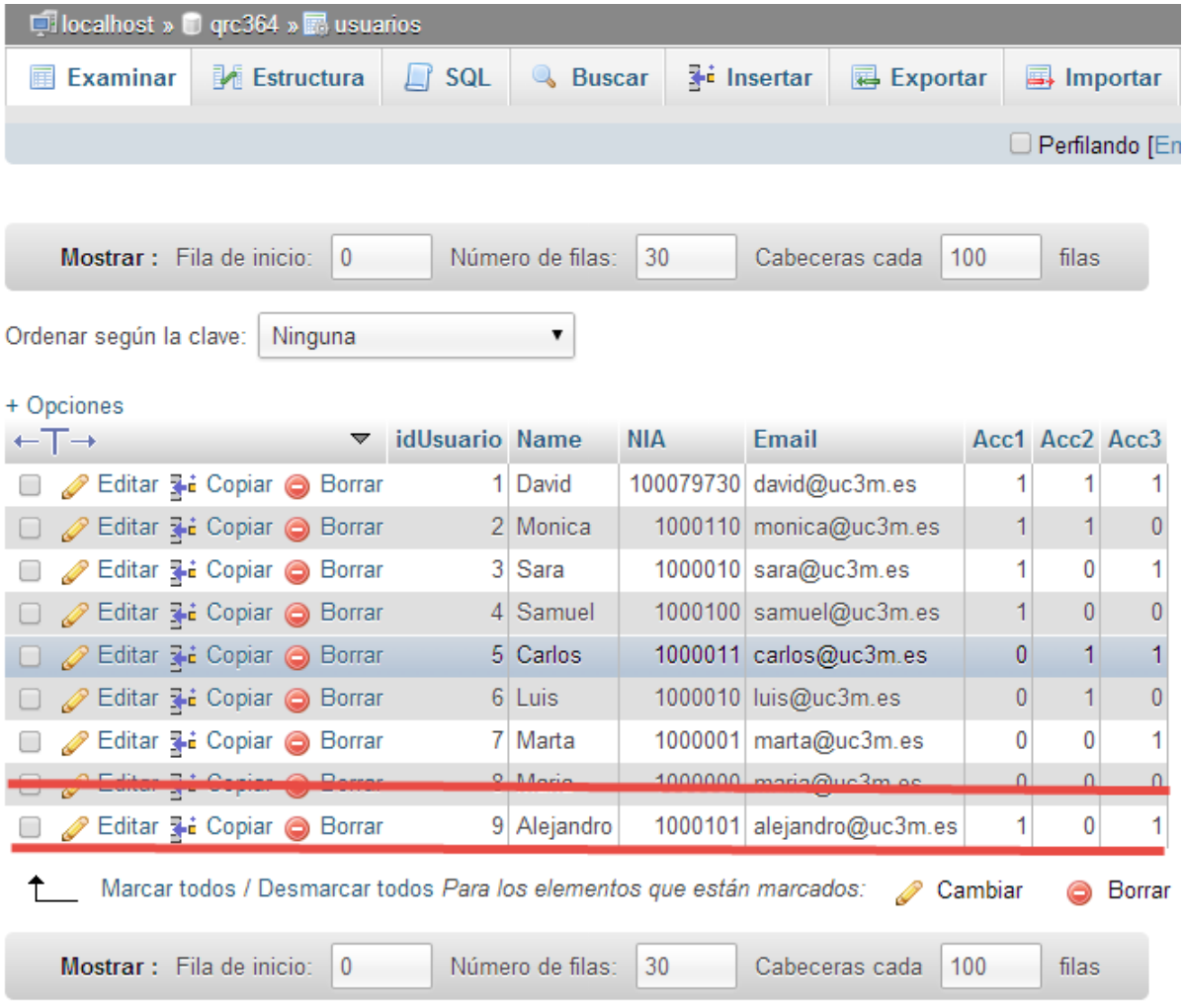
8.2 Verificaciones del sistema de accesos.

8.2.1 Comprobación de la correcta alta de un usuario con la aplicación Android.

Para llevar a cabo esta simulación se ha introducido los datos de un posible usuario en la interfaz de altas de usuario de la aplicación Android quedando esta pantalla con el siguiente resultado:



Una vez introducidos los datos y presionado el botón de "Guardar usuario" se puede comprobar la correcta introducción a través de la aplicación o con el uso de phpMyAdmin. A continuación se muestran las imágenes relativas a ambos resultados, mostrando en primer lugar la captura de pantalla de la aplicación Android y en segundo lugar la del sistema de gestión phpMyAdmin:

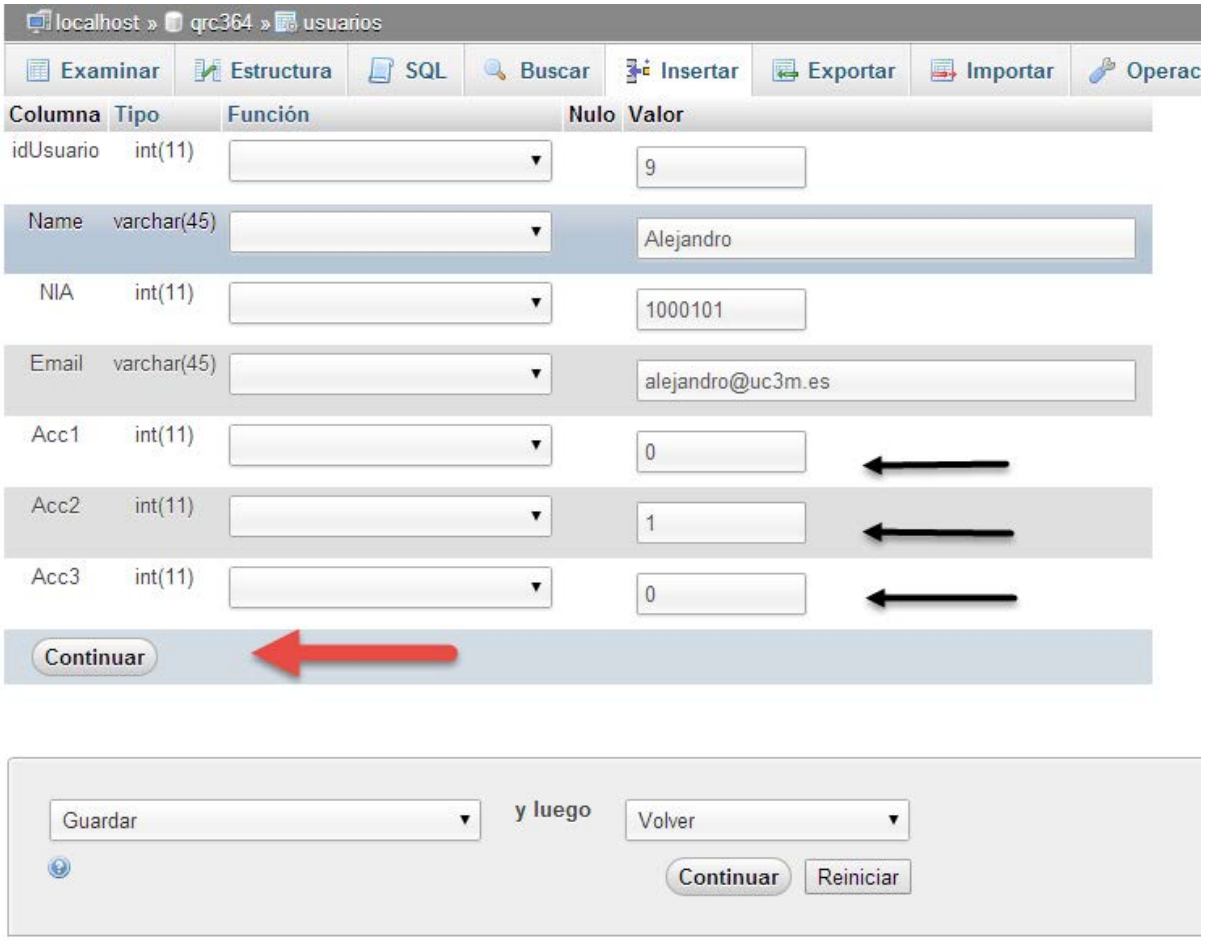


En ambos resultados se resalta en rojo que la introducción de datos ha sido efectuada con éxito.

8.2.2 Modificación de la información de un usuario con phpMyAdmin.

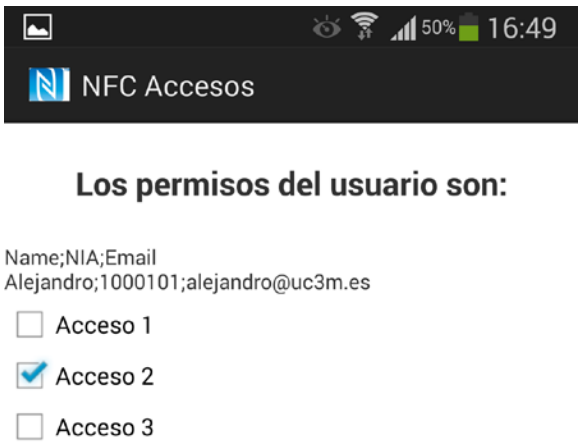
Para poder comprobar si es posible la modificación de un usuario a través del sistema de gestión de la base de datos, se han realizado cambios de permisos de accesos para el usuario introducido en la prueba anterior. De modo que este usuario tenga acceso sólo al área 2 (Acc2 = 1).

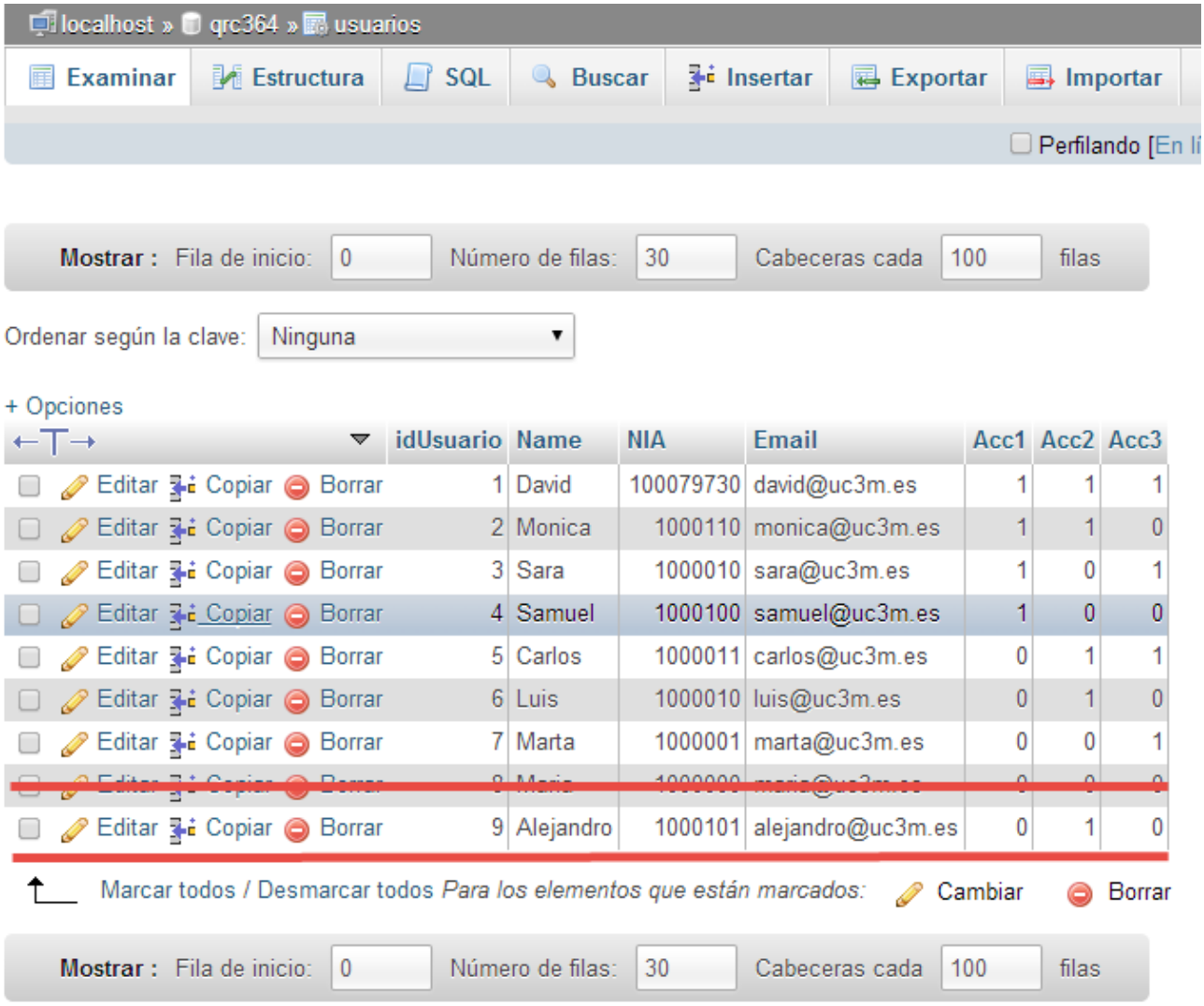
Este cambio se realiza pulsando “Editar” en la línea de información del usuario “Alejandro” y se realizan los cambios del siguiente modo:



En la imagen podemos comprobar que se ha efectuado el cambio manual de las 3 variables de acceso. Posteriormente para guardar estos cambios se presionará “Continuar”.

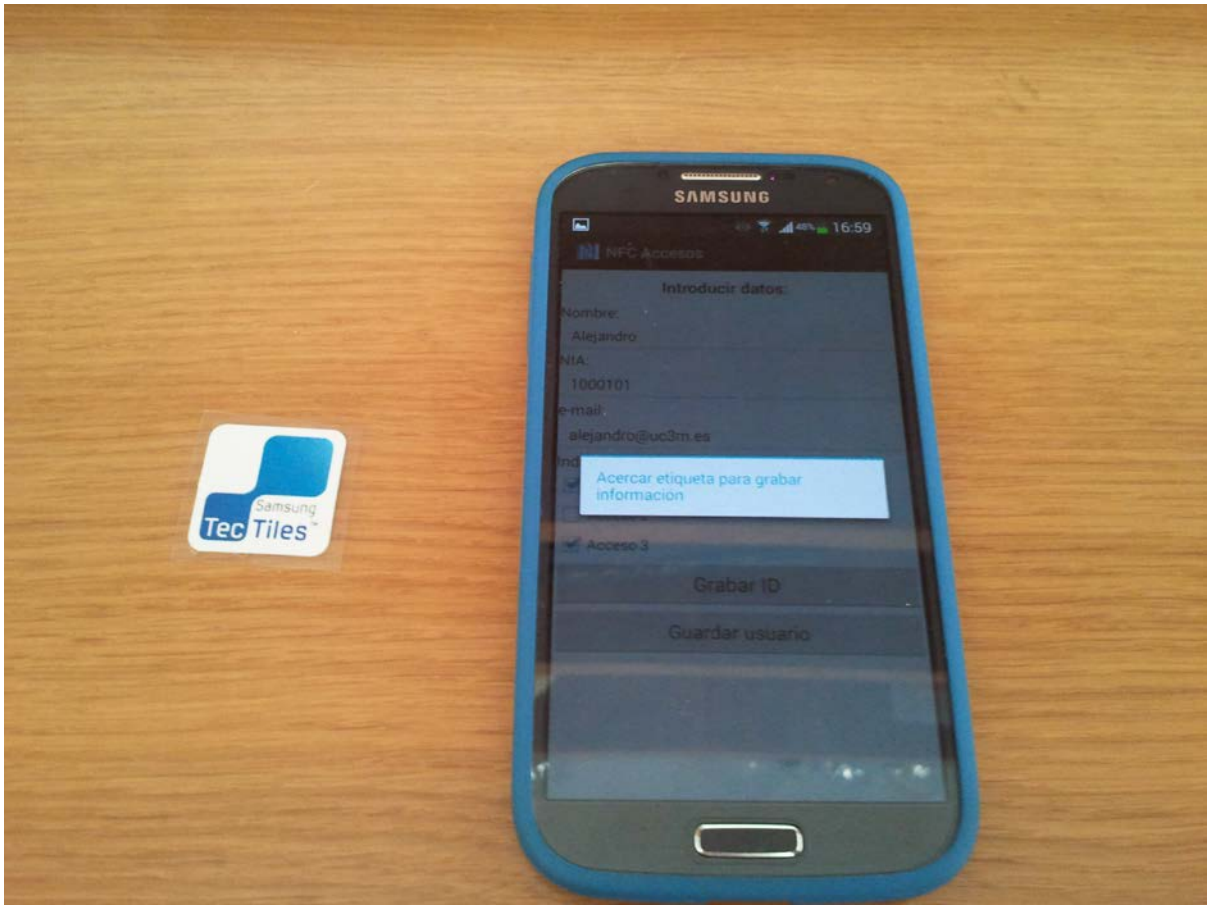
De este modo la información relativa al usuario "Alejandro" queda modificada, como comprobación mostramos en este caso la lectura de la etiqueta asociada a este usuario y posteriormente también mostramos el resultado en phpMyAdmin:



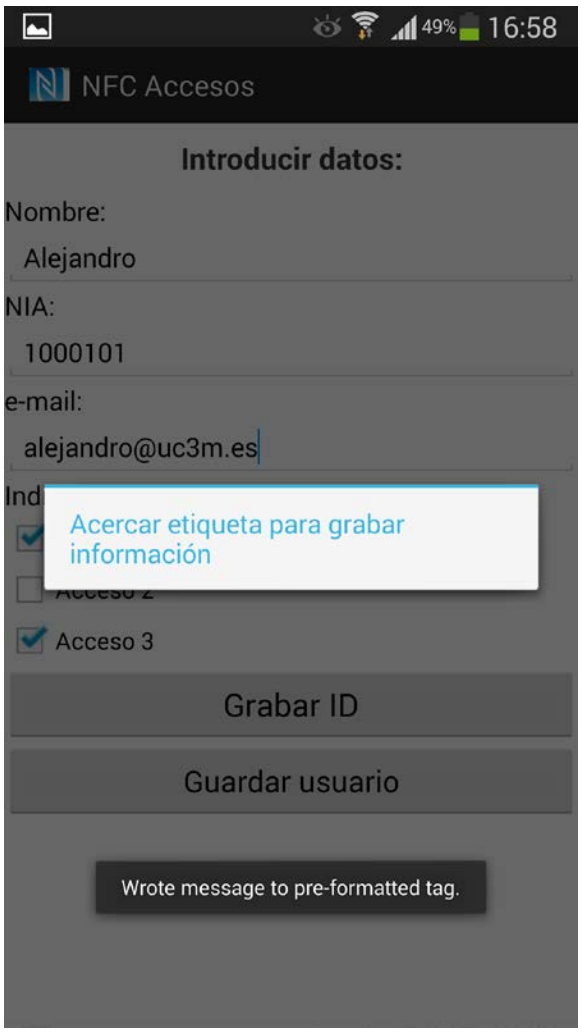

localhost » qrc364 » usuarios
Examinar
Estructura
SQL
Buscar
Insertar
Exportar
Importar
Perfilando [En l
Mostrar : Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas
Ordenar según la clave: Ninguna
+ Opciones
idUsuario Name NIA Email Acc1 Acc2 Acc3
Editar Copiar Borrar 1 David 100079730 david@uc3m.es 1 1 1
Editar Copiar Borrar 2 Monica 1000110 monica@uc3m.es 1 1 0
Editar Copiar Borrar 3 Sara 1000010 sara@uc3m.es 1 0 1
Editar Copiar Borrar 4 Samuel 1000100 samuel@uc3m.es 1 0 0
Editar Copiar Borrar 5 Carlos 1000011 carlos@uc3m.es 0 1 1
Editar Copiar Borrar 6 Luis 1000010 luis@uc3m.es 0 1 0
Editar Copiar Borrar 7 Marta 1000001 marta@uc3m.es 0 0 1
Editar Copiar Borrar 9 Alejandro 1000101 alejandro@uc3m.es 0 1 0
Marcar todos / Desmarcar todos Para los elementos que están marcados: Cambiar Borrar
Mostrar : Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

8.2.3 Grabación de un tag con el Smartphone.

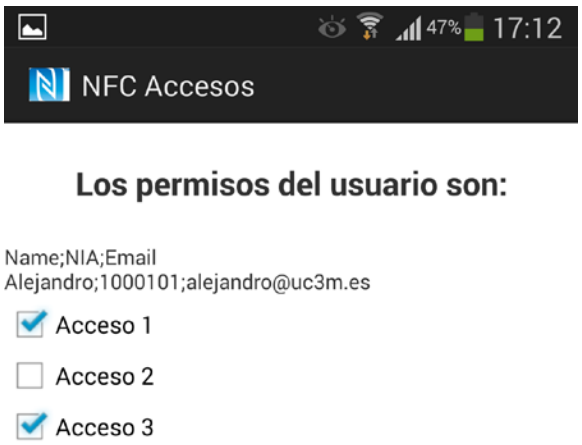
Para realizar la grabación de la información en un tag o etiqueta primero tendremos que completar la información correspondiente al usuario y después pulsar el botón de "Grabar ID" así el dispositivo mostrará un mensaje indicando que se acerque la etiqueta o tag. La siguiente imagen muestra cómo quedaría la aplicación tras pulsar el botón:



Después de acercar la etiqueta se habrá completado el grabado de esta cuando aparezca la ventana emergente informando de que se ha completado la escritura:



Además es posible comprobar la correcta escritura de la etiqueta leyendo esta con la aplicación y verificando así que el resultado que se obtiene de esta es el correcto, a continuación se muestra el resultado de leer esta etiqueta:

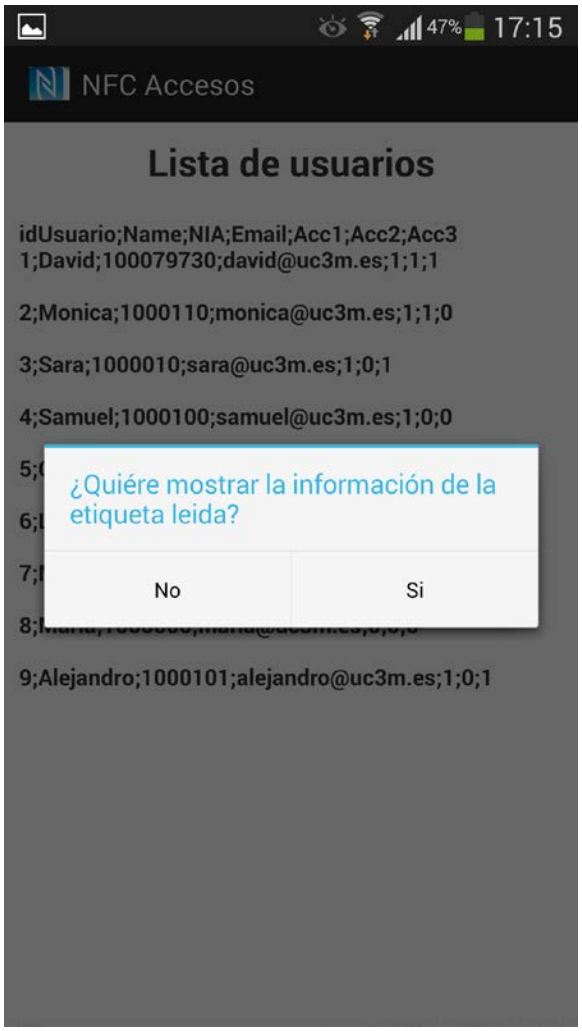


8.2.4 Lectura de un tag con el Smartphone con la aplicación en ejecución y en estado de reposo.

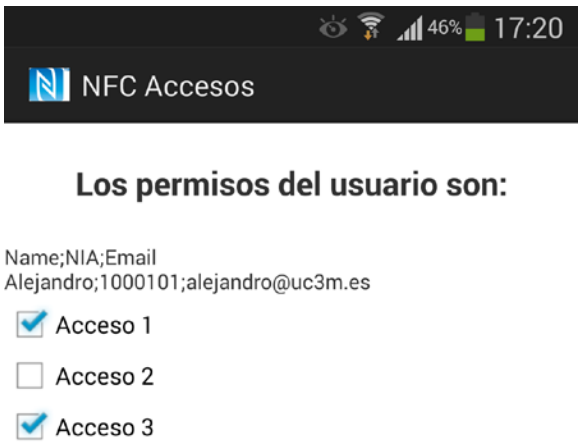
En el anterior apartado se comprobó el correcto funcionamiento de la lectura de una etiqueta con la aplicación en estado de reposo cuyo resultado arrojo la imagen anterior.

A continuación se verificará el correcto funcionamiento de la lectura de una etiqueta en el caso de que esta sea aproximada cuando la aplicación se encuentra en funcionamiento.

En este caso la aplicación, una vez acercada la etiqueta, pregunta al usuario si desea leer la etiqueta detectada o si por el contrario el usuario prefiere ignorar esta etiqueta y seguir viendo la pantalla en la que se encuentra situado.

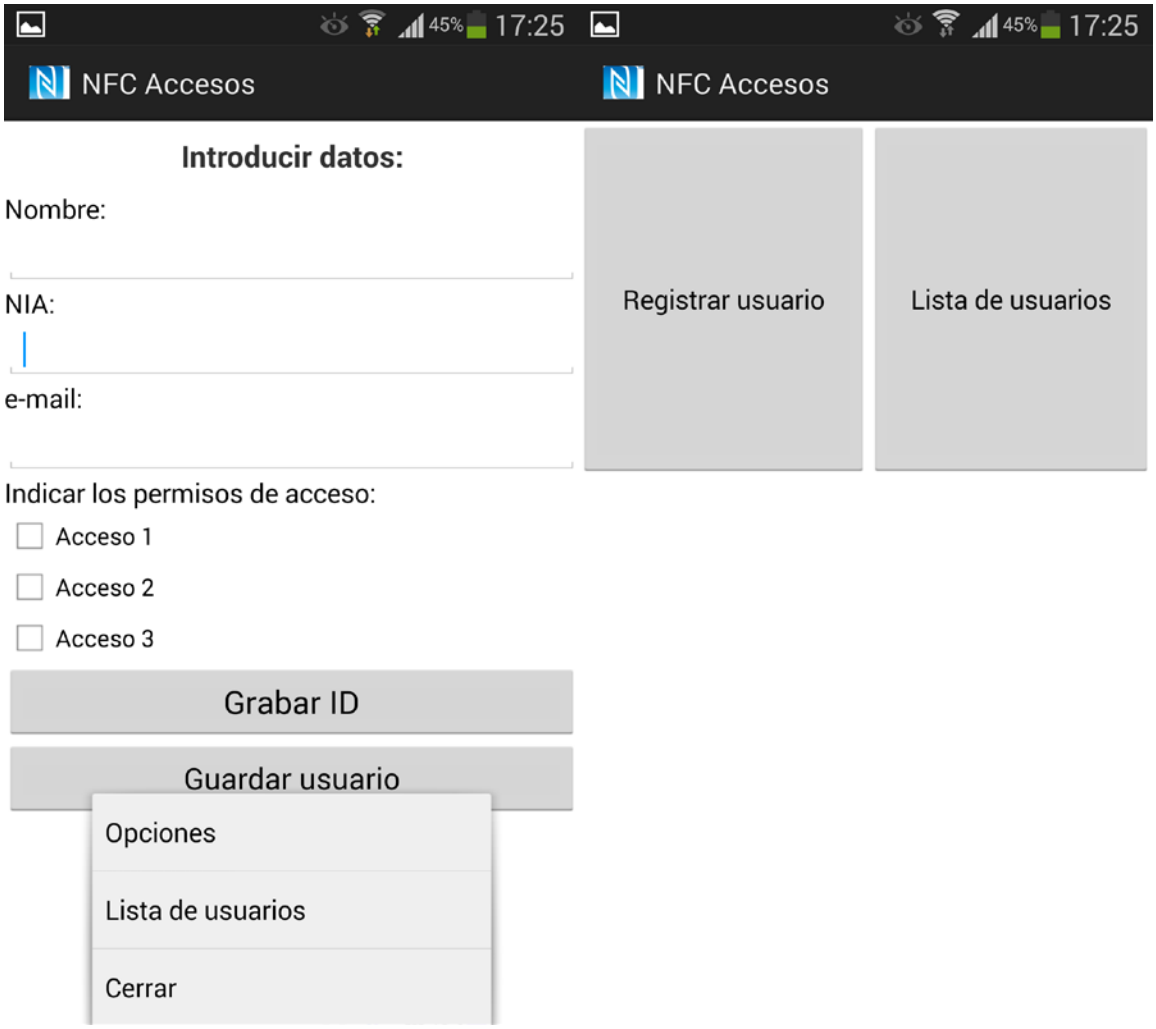


Para proceder con la verificación de la correcta lectura de la etiqueta NFC se pulsara afirmativamente a la cuestión aparecida en la ventana emergente. Comprobamos que el resultado es el esperado en la siguiente imagen:



8.2.5 Comprobación de contenido de la base de datos desde la aplicación Android.

Para llevar a cabo esta comprobación simplemente se procede pulsando el botón de "Lista de usuarios" desde la aplicación Android, esto es posible desde el menú o desde la pantalla de opciones. Ambas opciones se muestran a continuación:



Como resultado de pulsar en estos botones la aplicación nos envía a la pantalla donde comprobamos que efectivamente los usuarios han sido dados de alta satisfactoriamente. La siguiente imagen muestra esta comprobación:


45%
17:28
NFC Accesos
Lista de usuarios
idUsuario;Name;NIA;Email;Acc1;Acc2;Acc3
1;David;100079730;david@uc3m.es;1;1;1
2;Monica;1000110;monica@uc3m.es;1;1;0
3;Sara;1000010;sara@uc3m.es;1;0;1
4;Samuel;1000100;samuel@uc3m.es;1;0;0
5;Carlos;1000011;carlos@uc3m.es;0;1;1
6;Luis;1000010;luis@uc3m.es;0;1;0
7;Marta;1000001;marta@uc3m.es;0;0;1
8;Maria;1000000;maria@uc3m.es;0;0;0
9;Alejandro;1000101;alejandro@uc3m.es;1;0;1

9 Conclusiones.

El desarrollo de esta aplicación Android ha cumplido con todas las expectativas que tenía puestas en este proyecto de Android, que eran conocer con más detalle el funcionamiento de un sistema operativo cada día más presente en la vida de todos y con un gran futuro por delante.

Para poder llegar a la conclusión de este proyecto también se utilizaron otras características de Android, como el uso de bases de datos específicas e internas de este sistema operativo, como es el caso de SQLite. Aunque posteriormente este tipo de forma de trabajo fue sustituido con el actual uso de la base de datos externas, ha resultado muy instructivo a la hora de comprender como funcionan las bases de datos de una forma más básica.

9.1 Análisis de los resultados obtenidos.

En este proyecto se ha creado un sistema completamente operativo y funcional que podría aplicarse, aunque también hay que tener en cuenta sus limitaciones y sus necesarias ampliaciones para ser completamente efectivo para su empleo en un caso real.

Las mejoras que pueden aplicarse a este proyecto para su implementación en caso real son los siguientes:

- El uso de tag o etiquetas NFC que sólo permitan la lectura o su manipulación con un hardware especifico que impida a cualquier usuario la suplantación de otro.
- Niveles de permiso para el tratamiento de la base de datos. En esta aplicación existe la posibilidad de realizar altas de usuario en la base de datos de modo que cualquier usuario puede auto-asignarse todos los privilegios de acceso.
- En este sistema desarrollado se podría producir el robo de identidad en caso de sustracción de etiqueta identificadora.
- También entra en juego el factor humano que puede ocasionar errores de interpretación, cómo una lectura errónea de los datos del usuario leído.

9.2 Futuras aplicaciones.

Cabe destacar que este trabajo puede dar pie a la creación de futuras ampliaciones con el uso de la tecnología NFC, no sólo como el control de accesos si no que podría servir, por ejemplo, usando una base de datos más elaborada con la que podríamos mostrar toda la información relevante de un alumno en la Universidad Carlos III. También pueden usarse estos principios para la apertura de cerraduras con Smartphone o con una etiqueta, el control de libros de una biblioteca, el acceso a los diferentes transportes públicos, etc. En definitiva las tecnologías empleadas en este proyecto tienen un gran futuro por sus grandes posibilidades, además de ser una tecnología barata ya que las etiquetas tienen precios muy económicos y día a día los dispositivos móviles tienden a incorporar la tecnología NFC entre sus características.

10 Bibliografía

[1]

Mobile phone. [Online] http://en.wikipedia.org/wiki/Mobile_phone.

History of mobile phones. [Online] http://en.wikipedia.org/wiki/History_of_mobile_phones.

[2]

Smartphone. [Online] http://en.wikipedia.org/wiki/Smartphone

[3]

Open Hadset Alliance. [Online] http://es.wikipedia.org/wiki/Open_Handset_Alliance.

[4]

Android Developers. [Online] http://developer.android.com/develop/index.html.

[5]

iOS. [Online] http://es.wikipedia.org/wiki/IOS.

App Store. [Online] http://es.wikipedia.org/wiki/App_Store.

[6]

Windows Phone. [Online] http://es.wikipedia.org/wiki/Windows_Phone.

[7]

BlackBerry Developer Activities. [Online] http://en.wikipedia.org/wiki/BlackBerry_10#Developer_activities.

BlackBerry. [Online] http://en.wikipedia.org/wiki/BlackBerry.

[8]

NFC Forum. [Online] http://nfc-forum.org/.

[9]

NFC Forum: Technical Specifications. [Online] http://members.nfc-forum.org/specs/spec_list/.

[10]

G.Taylor, Allen. *SQL for Dummies.*

Database. [En línea] http://en.wikipedia.org/wiki/Database.

[11]

Modelo relacional. [En línea] http://es.wikipedia.org/wiki/Modelo_de_base_de_datos#Modelo_relacional.

[12]

Object-relational impedance mismatch. [En línea] http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch.

[13]

Android Developers. [En línea] http://developer.android.com/about/index.html.

[14]

Developers Tools. [En línea] http://developer.android.com/tools/index.html.

[15]

MySQL Reference Manual. [En línea] https://dev.mysql.com/doc/refman/5.0/es/index.html.

[16]

phpMyAdmin [En línea] http://www.phpmyadmin.net/home_page/index.php

11 Anexos.

11.1 Código de la aplicación.

11.1.1 MainActivity.java

```
package com.example.nfctagrecord;

import java.io.IOException;

public class MainActivity extends Activity {
    private static final String TAG = "nfctagrecord";
    private boolean mResumed = false;
    private boolean mWriteMode = false;
    NfcAdapter mNfcAdapter;
    EditText mNote;

    // BASE DE DATOS EXT.
    String textIP, textPuerto, textContrasena, textUsuario;
    String catalogoMySQL;
    static Connection conexionMySQL;
    static String SQLEjecutar;
    TextView textResultadoSQL;
    String ipServidorMySQL, contrasenaMySQL, usuarioMySQL, puertoMySQL;

    // Varibles nueva versión
    String name;
    String tag;
    String acc;
    String acc1= "0";
    String acc2= "0";
    String acc3= "0";
    EditText mName;
    EditText mNIA;

    // Check variables
    public CheckBox check1;
    public CheckBox check2;
    public CheckBox check3;

    public CheckBox check1tag;
    public CheckBox check2tag;
    public CheckBox check3tag;


    PendingIntent mNfcPendingIntent;
    IntentFilter[] mWriteTagFilters;
    IntentFilter[] mNdefExchangeFilters;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mNfcAdapter = NfcAdapter.getDefaultAdapter(this);

        setContentView(R.layout.new_tag);

        mNIA = ((EditText) findViewById(R.id.editNIA));
        mNIA.addTextChangedListener(mTextWatcher);


        // Maneja todos los NFC intents recividos en esta actividad.
        mNfcPendingIntent = PendingIntent.getActivity(this, 0, new Intent(this,
                getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
```

```
// Intent filters for reading a note from a tag or exchanging over p2p.
IntentFilter ndefDetected = new IntentFilter(
        NfcAdapter.ACTION_NDEF_DISCOVERED);
try {
    ndefDetected.addDataType("text/plain");
} catch (MalformedMimeTypeException e) {
}
mNdefExchangeFilters = new IntentFilter[] { ndefDetected };

// Intent filters for writing to a tag
IntentFilter tagDetected = new IntentFilter(
        NfcAdapter.ACTION_TAG_DISCOVERED);
mWriteTagFilters = new IntentFilter[] { tagDetected };

// Inicialización de CheckBottons
check1 = (CheckBox) findViewById(R.id.checkBox01);
check1.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {

        if (isChecked) {
            toast("Checkbox marcado!");
            acc1 = "1";
        } else {
            toast("Checkbox desmarcado!");
            acc1 = "0";
        }
    }
});
check2 = (CheckBox) findViewById(R.id.checkBox02);
check2.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {

        if (isChecked) {
            toast("Checkbox marcado!");
            acc2 = "1";
        } else {
            toast("Checkbox desmarcado!");
            acc2 = "0";
        }
    }
});
check3 = (CheckBox) findViewById(R.id.checkBox03);
check3.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {

        if (isChecked) {
            toast("Checkbox marcado!");
            acc3 = "1";
        } else {
            toast("Checkbox desmarcado!");
            acc3 = "0";
        }
    }
```

```
        });

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);

        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // TODO Auto-generated method stub
        switch (item.getItemId()) {
        case R.id.close:
            finish();
            break;
        case R.id.action_settings:
            setContentView(R.layout.activity_main);
            break;
        case R.id.list:
            setContentView(R.layout.result);

            TextView list = (TextView) findViewById(R.id.list);
            list.setText(name);

            // BASE DE DATOS EXT.
            cargarConfiguracion();
            SQLEjecutar = "select * from usuarios";
            conectarBDMySQL(usuarioMySQL, contrasenaMySQL, ipServidorMySQL, puertoMySQL,
catalogoMySQL);
            String resultadoSQL = ejecutarConsultaSQL(false, "insert into usuarios values (null,
'paula', 'd1114d6', 'cvm@htaa.es', '0','1','0' )");
            TextView textResultadoSQL = (TextView) findViewById(R.id.tag_permission_user);
            textResultadoSQL.setText(resultadoSQL);

            break;

        }

        return false;
    }

    public void onClose(View botton) {
        finish();
    }

    @Override
    protected void onResume() {
        super.onResume();
        mResumed = true;
        // NFCtagrecord received from Android
        if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {
            NdefMessage[] messages = getNdefMessages(getIntent());
            byte[] payload = messages[0].getRecords()[0].getPayload();
```

```
            setNoteBody(new String(payload));

            String bNIA = new String(messages[0].getRecords()[0].getPayload());

            setContentView(R.layout.tag_permissions);

            String buscar="select Name, NIA, email from usuarios where NIA='"+bNIA+"';";
            cargarConfiguracion();
            SQLEjecutar =buscar;
            conectarBDMySQL(usuarioMySQL, contrasenaMySQL, ipServidorMySQL, puertoMySQL,
catalogoMySQL);
            String resultadoSQL = ejecutarConsultaSQL(false, buscar);
            TextView textResultadoSQL = (TextView) findViewById(R.id.tag_permission_user);
            textResultadoSQL.setText(resultadoSQL);


            String comprobar1 = "select Acc1 from usuarios where NIA='"+bNIA+"';";
            String comprobar2 = "select Acc2 from usuarios where NIA='"+bNIA+"';";
            String comprobar3 = "select Acc3 from usuarios where NIA='"+bNIA+"';";

            SQLEjecutar = comprobar1;
            String Acc1result= ejecutarConsultaSQL(false, comprobar1);
            check1tag = (CheckBox) findViewById(R.id.checkBox1tag);

            SQLEjecutar = comprobar2;
            String Acc2result= ejecutarConsultaSQL(false, comprobar2);
            check2tag = (CheckBox) findViewById(R.id.checkBox2tag);

            SQLEjecutar = comprobar3;
            String Acc3result= ejecutarConsultaSQL(false, comprobar3);
            check3tag = (CheckBox) findViewById(R.id.checkBox3tag);

            int aux1 = lectorString (Acc1result, 2);
            int aux2 = lectorString (Acc2result, 1);
            int aux3 = lectorString (Acc3result, 1);

            check1tag.setChecked(check(aux1));
            check2tag.setChecked(check(aux2));
            check3tag.setChecked(check(aux3));

            setIntent(new Intent()); // Consume this intent.
        }
        enableNdefExchangeMode();
    }

    boolean check (int aux){
        if (aux == 1){
            return true;

        }else{
            return false;

        }
    }

    int lectorString (String ACC, int m){
```

```
    int count = 0;
    for (int i=0; i<7;i++){
        if (ACC.charAt(i) == '1'){
            count++;
            if (count == m){
                return 1;
            }
        }
    }
    return 0;
}


@Override
protected void onPause() {
    super.onPause();
    mResumed = false;
    mNfcAdapter.disableForegroundNdefPush(this);
}

@Override
protected void onNewIntent(Intent intent) {
    // Modo lectura NDEF
    if (!mWriteMode
            && NfcAdapter.ACTION_NDEF_DISCOVERED.equals(intent.getAction())) {
        NdefMessage[] msgs = getNdefMessages(intent);
        promptForContent(msgs[0]);

    }

    // Modo lectura
    if (mWriteMode
            && NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
        Tag detectedTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
        writeTag(getNoteAsNdef(), detectedTag);

    }
}

private TextWatcher mTextWatcher = new TextWatcher() {

    @Override
    public void onTextChanged(CharSequence mNIA, int arg1, int arg2,
            int arg3) {

    }

    @Override
    public void beforeTextChanged(CharSequence mNIA, int arg1, int arg2,
            int arg3) {

    }

    @Override
    public void afterTextChanged(Editable mNIA) {
        if (mResumed) {
```

```
            mNfcAdapter.enableForegroundNdefPush(MainActivity.this,
                    getNoteAsNdef());
        }
    }
};

public void onNewClick(View botton) {

    setContentView(R.layout.new_tag);

    check1 = (CheckBox) findViewById(R.id.checkBox01);
    check1.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked) {

            if (isChecked) {
                toast("Checkbox marcado!");
                acc1 = "1";
            } else {
                toast("Checkbox desmarcado!");
                acc1 = "0";
            }
        }
    });
    check2 = (CheckBox) findViewById(R.id.checkBox02);
    check2.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked) {

            if (isChecked) {
                toast("Checkbox marcado!");
                acc2 = "1";
            } else {
                toast("Checkbox desmarcado!");
                acc2 = "0";
            }
        }
    });
    check3 = (CheckBox) findViewById(R.id.checkBox03);
    check3.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked) {

            if (isChecked) {
                toast("Checkbox marcado!");
                acc3 = "1";
            } else {
                toast("Checkbox desmarcado!");
                acc3 = "0";
            }
        }
    });

}

public void onSave(View botton) {
```

```
        TextView sname = (TextView) findViewById(R.id.editName);
        TextView snia = (TextView) findViewById(R.id.editNIA);
        TextView semail = (TextView) findViewById(R.id.editemail);
        insertTag(sname.getText().toString(), snia.getText().toString(), acc1,
                acc2, acc3, semail.getText().toString());

    }

    public void onListClick(View botton){
        setContentView(R.layout.result);

        TextView list = (TextView) findViewById(R.id.list);
        list.setText(name);

        // BASE DE DATOS EXT.
        cargarConfiguracion();
        SQLEjecutar = "select * from usuarios";
        conectarBDMySQL(usuarioMySQL, contrasenaMySQL, ipServidorMySQL, puertoMySQL,
catalogoMySQL);
        String resultadoSQL = ejecutarConsultaSQL(false, "insert into usuarios values (null
'paula', 'd1114d6', 'cvm@htaa.es', '0','1','0' )");
        TextView textResultadoSQL = (TextView) findViewById(R.id.tag_permission_user);
        textResultadoSQL.setText(resultadoSQL);

    }

    public void onRecordTag(View botton) {
        // Write to a tag for as long as the dialog is shown.
        disableNdefExchangeMode();
        enableTagWriteMode();

        new AlertDialog.Builder(MainActivity.this)
                .setTitle("Acercar etiqueta para grabar información")
                .setOnCancelListener(new DialogInterface.OnCancelListener() {
                    @Override
                    public void onCancel(DialogInterface dialog) {
                        disableTagWriteMode();
                        enableNdefExchangeMode();
                    }
                }).create().show();
    }

    private void promptForContent(final NdefMessage msg) {
        new AlertDialog.Builder(this)
                .setTitle(
                        "¿Quiére mostrar la información de la etiqueta leida?")
                .setPositiveButton("Si", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface arg0, int arg1) {
                        String body = new String(msg.getRecords()[0]
                                .getPayload());
                        setNoteBody(body);
                        tag = body;

                        String bNIA = body;

                        setContentView(R.layout.tag_permissions);
```

```
                        String buscar = "select Name, NIA, email from usuarios where NIA='"
                                + bNIA + "';";
                        cargarConfiguracion();
                        SQLEjecutar = buscar;
                        conectarBDMySQL(usuarioMySQL, contrasenaMySQL,
                                ipServidorMySQL, puertoMySQL, catalogoMySQL);
                        String resultadoSQL = ejecutarConsultaSQL(false, buscar);
                        TextView textResultadoSQL = (TextView)
findViewById(R.id.tag_permission_user);
                        textResultadoSQL.setText(resultadoSQL);

                        String comprobar1 = "select Acc1 from usuarios where NIA='"
                                + bNIA + "';";
                        String comprobar2 = "select Acc2 from usuarios where NIA='"
                                + bNIA + "';";
                        String comprobar3 = "select Acc3 from usuarios where NIA='"
                                + bNIA + "';";

                        SQLEjecutar = comprobar1;
                        String Acc1result = ejecutarConsultaSQL(false,
                                comprobar1);
                        check1tag = (CheckBox) findViewById(R.id.checkBox1tag);

                        SQLEjecutar = comprobar2;
                        String Acc2result = ejecutarConsultaSQL(false,
                                comprobar2);
                        check2tag = (CheckBox) findViewById(R.id.checkBox2tag);

                        SQLEjecutar = comprobar3;
                        String Acc3result = ejecutarConsultaSQL(false,
                                comprobar3);
                        check3tag = (CheckBox) findViewById(R.id.checkBox3tag);

                        int aux1 = lectorString(Acc1result, 2);
                        int aux2 = lectorString(Acc2result, 1);
                        int aux3 = lectorString(Acc3result, 1);

                        check1tag.setChecked(check(aux1));
                        check2tag.setChecked(check(aux2));
                        check3tag.setChecked(check(aux3));

                        setIntent(new Intent()); // Consume this intent.

                    }
                })
                .setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface arg0, int arg1) {

                    }
                }).show();
    }

    private void setNoteBody(String body) {
        Editable text = mNIA.getText();
        text.clear();
```

```
    text.append(body);



}

private NdefMessage getNoteAsNdef() {
    byte[] textBytes = mNIA.getText().toString().getBytes();
    NdefRecord textRecord = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
            "text/plain".getBytes(), new byte[] {}, textBytes);
    return new NdefMessage(new NdefRecord[] { textRecord });
}

NdefMessage[] getNdefMessages(Intent intent) {
    // Parse the intent
    NdefMessage[] msgs = null;
    String action = intent.getAction();
    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action)
            || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
        Parcelable[] rawMsgs = intent
                .getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
        if (rawMsgs != null) {
            msgs = new NdefMessage[rawMsgs.length];
            for (int i = 0; i < rawMsgs.length; i++) {
                msgs[i] = (NdefMessage) rawMsgs[i];
            }
        } else {
            // Unknown tag type
            byte[] empty = new byte[] {};
            NdefRecord record = new NdefRecord(NdefRecord.TNF_UNKNOWN,
                    empty, empty, empty);
            NdefMessage msg = new NdefMessage(new NdefRecord[] { record });
            msgs = new NdefMessage[] { msg };
        }
    } else {
        Log.d(TAG, "Unknown intent.");
        finish();
    }
    return msgs;
}

private void enableNdefExchangeMode() {
    mNfcAdapter
            .enableForegroundNdefPush(MainActivity.this, getNoteAsNdef());
    mNfcAdapter.enableForegroundDispatch(this, mNfcPendingIntent,
            mNdefExchangeFilters, null);
}

private void disableNdefExchangeMode() {
    mNfcAdapter.disableForegroundNdefPush(this);
    mNfcAdapter.disableForegroundDispatch(this);
}

private void enableTagWriteMode() {
    mWriteMode = true;
    IntentFilter tagDetected = new IntentFilter(
            NfcAdapter.ACTION_TAG_DISCOVERED);
```

```
    mWriteTagFilters = new IntentFilter[] { tagDetected };
    mNfcAdapter.enableForegroundDispatch(this, mNfcPendingIntent,
            mWriteTagFilters, null);
}

private void disableTagWriteMode() {
    mWriteMode = false;
    mNfcAdapter.disableForegroundDispatch(this);
}

boolean writeTag(NdefMessage message, Tag tag) {
    int size = message.toByteArray().length;

    try {
        Ndef ndef = Ndef.get(tag);
        if (ndef != null) {
            ndef.connect();

            if (!ndef.isWritable()) {
                toast("Tag de sólo lectura.");
                return false;
            }
            if (ndef.getMaxSize() < size) {
                toast("La capacidad del tag es " + ndef.getMaxSize()
                        + " bytes, el mensaje es " + size + " bytes.");
                return false;
            }

            ndef.writeNdefMessage(message);
            toast("Mensaje escrito satisfactoriamente.");
            return true;
        } else {
            NdefFormatable format = NdefFormatable.get(tag);
            if (format != null) {
                try {
                    format.connect();
                    format.format(message);
                    toast("Tag formateada y mensaje escrito");
                    return true;
                } catch (IOException e) {
                    toast("Error en el formateo.");
                    return false;
                }
            } else {
                toast("La etiqueta no soporta NDEF.");
                return false;
            }
        }
    } catch (Exception e) {
        toast("Error en la escritura de la etiqueta");
    }

    return false;
}

private void toast(String text) {
    Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
```

```
    }

    // INTRODUCCIÓN DATOS, BUSQUEDA DATOS EN LA BASE DE DATOS EXTERNA

    public void insertTag(String name, String tag, String acc1, String acc2,
            String acc3, String email) {

        String insert = "insert into usuarios values(NULL, '"
                + name
                + "','"
                + tag
                + "', '"
                + email
                + "', '"
                + acc1
                + "', '"
                + acc2
                + "', '"
                + acc3 +"')";
        cargarConfiguracion();
        SQLEjecutar = insert;
        conectarBDMySQL(usuarioMySQL, contrasenaMySQL, ipServidorMySQL, puertoMySQL,
catalogoMySQL);
        ejecutarConsultaSQL(true, insert);

    }



    //CONEXIÓN BASE DE DATOS EXT.

    public void cargarConfiguracion() {
        // leemos los valores de conexiÃ³n al servidor
        // MySQL desde SharedPreferences
        catalogoMySQL = [redacted];
        ipServidorMySQL = [redacted];
        contrasenaMySQL = [redacted];
        puertoMySQL = [redacted];
        usuarioMySQL = [redacted];
    }

    public static String ejecutarConsultaSQL(Boolean SQLModificacion, String context)
    {
        try
        {
            String resultadoSQL = "";
            //ejecutamos consulta SQL de selecciÃ³n (devuelve datos)
            if (!SQLModificacion)
            {
                Statement st = conexionMySQL.createStatement();
                ResultSet rs = st.executeQuery(SQLEjecutar);

                Integer numColumnas = 0;

                //nÃºmero de columnas (campos) de la consula SQL
                numColumnas = rs.getMetaData().getColumnCount();
```

```
        //obtenemos el tÃ-tulo de las columnas
        for (int i = 1; i <= numColumnas; i++)
        {
            if (resultadoSQL != "")
                if (i < numColumnas)
                    resultadoSQL = resultadoSQL +
                    rs.getMetaData().getColumnName(i).toString() + ";";
                else
                    resultadoSQL = resultadoSQL +
                    rs.getMetaData().getColumnName(i).toString();
            else
                if (i < numColumnas)
                    resultadoSQL =
                    rs.getMetaData().getColumnName(i).toString() + ";";
                else
                    resultadoSQL =
                    rs.getMetaData().getColumnName(i).toString();
        }


        //mostramos el resultado de la consulta SQL
        while (rs.next())
        {
            resultadoSQL = resultadoSQL + "\n";

            //obtenemos los datos de cada columna
            for (int i = 1; i <= numColumnas; i++)
            {
                if (rs.getObject(i) != null)
                {
                    if (resultadoSQL != "")
                        if (i < numColumnas)
                            resultadoSQL = resultadoSQL +
                            rs.getObject(i).toString() + ";";
                        else
                            resultadoSQL = resultadoSQL +
                            rs.getObject(i).toString();
                    else
                        if (i < numColumnas)
                            resultadoSQL = rs.getObject(i).toString() + ";";
                        else
                            resultadoSQL = rs.getObject(i).toString();
                }
                else
                {
                    if (resultadoSQL != "")
                        resultadoSQL = resultadoSQL + "null;";
                    else
                        resultadoSQL = "null;";
                }
            }
            resultadoSQL = resultadoSQL + "\n";
        }
        st.close();
        rs.close();
    }
    // consulta SQL de modificaciÃ³n de
```

```
        // datos (CREATE, DROP, INSERT, UPDATE)
        else
        {
            int numAfectados = 0;
            Statement st = conexionMySQL.createStatement();
            numAfectados = st.executeUpdate(SQLEjecutar);
            resultadoSQL = "Registros afectados: " + String.valueOf(numAfectados);
            st.close();
        }
        return resultadoSQL;
    }

    catch (Exception e)
    {

        return "";
    }
}

public void conectarBDMySQL(String usuario, String contrasena, String ip,
        String puerto, String catalogo) {
    if (usuario == "" || puerto == "" || ip == "") {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(
                MainActivity.this);
        alertDialog.setMessage("Antes de establecer la conexiÃ³n "
                + "con el servidor "
                + "MySQL debe indicar los datos de conexiÃ³n "
                + "(IP, puerto, usuario y contraseÃ±a).");
        alertDialog.setTitle("Datos conexiÃ³n MySQL");
        alertDialog.setIcon(android.R.drawable.ic_dialog_alert);
        alertDialog.setCancelable(false);
        alertDialog.setPositiveButton("Aceptar",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        }
                });
        alertDialog.show();
    } else {
        String urlConexionMySQL = "";
        if (catalogo != "")
            urlConexionMySQL = "jdbc:mysql://" + ip + ":" + puerto + "/"
                    + catalogo;
        else
            urlConexionMySQL = "jdbc:mysql://" + ip + ":" + puerto;
        if (usuario != "" & contrasena != "" & ip != "" & puerto != "") {
            try {
                Class.forName("com.mysql.jdbc.Driver");
                conexionMySQL = DriverManager.getConnection(
                        urlConexionMySQL, usuario, contrasena);
            } catch (ClassNotFoundException e) {
                Toast.makeText(getApplicationContext(),
                        "Error: " + e.getMessage(), Toast.LENGTH_SHORT)
                        .show();
            } catch (SQLException e) {
                Toast.makeText(getApplicationContext(),
                        "Error: " + e.getMessage(), Toast.LENGTH_SHORT)
                        .show();
```

```
                }
            }
        }
    }

}
```

11.1.2 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.nfctagrecord"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="9" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.NFC"/>


    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.nfctagrecord.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <!-- Maneja las notas detectadas de fuera de la aplicacion  -->
            <intent-filter>
                <action android:name="android.nfc.action.NDEF_DISCOVERED" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="text/plain" />
            </intent-filter>
        </activity>

        <activity
        android:name=".Result" android:label="@string/tag_list">
        </activity>

        <activity
        android:name=".DatabaseHelper" android:label="@string/DatabaseHelper">
        </activity>

    </application>

</manifest>
```

11.1.3 Strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">NFC Accesos</string>
    <string name="write_to_tag">Registrar usuario</string>
    <string name="read_tag">Read Tag</string>
    <string name="default_note_text">Edit me!</string>
    <string name="action_settings">Opciones</string>
    <string name="hello_world">Hello world!</string>

    <string name="tag_list">Lista de usuarios</string>
    <string name="show_list">Lista de usuarios</string>
    <string name="list"></string>
    <string name="close">Cerrar</string>

    <string name="listdb"></string>
    <string name="DatabaseHelper">DatabaseHelper</string>

    <string name="new_tag">Introducir datos:</string>

    <string name="name">Nombre:</string>

    <string name="nia">NIA:</string>

    <string name="save">Guardar usuario</string>

    <string name="default_name"></string>

    <string name="default_nia"></string>

    <string name="email">e-mail:</string>

    <string name="default_email"></string>

    <string name="permission">Indicar  los permisos de acceso:</string>

    <string name="acc1">Acceso 1</string>

    <string name="acc2">Acceso 2</string>

    <string name="acc3">Acceso 3</string>

    <string name="delete_database">Borrar base de datos</string>

    <string name="recordTag">Grabar ID</string>

    <string name="user_permissions">Los permisos del usuario son:</string>


</resources>
```

11.1.4 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="223dp"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/read_tag"
            android:layout_width="192dp"
            android:layout_height="match_parent"
            android:layout_weight="0.88"
            android:onClick="onNewClick"
            android:text="@string/write_to_tag" />

        <Button
            android:id="@+id/show_list"
            android:layout_width="188dp"
            android:layout_height="match_parent"
            android:layout_weight="0.86"
            android:onClick="onListClick"
            android:text="@string/show_list" />

    </LinearLayout>

    <Button
        android:id="@+id/Button01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onDeleteDB"
        android:text="@string/delete_database" />

</LinearLayout>
```

11.1.5 new_tag.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:focusableInTouchMode="false"
        android:gravity="center"
        android:padding="10sp"
        android:text="@string/new_tag"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/name"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/editName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:text="@string/default_name" />

    <TextView
        android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/nia"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/editNIA"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberSigned"
        android:text="@string/default_nia" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/TextView03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/email"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/editemail"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textWebEmailAddress"
            android:text="@string/default_email" />

        <TextView
            android:id="@+id/TextView04"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/permission"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <CheckBox
            android:id="@+id/checkBox01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/acc1" />

        <CheckBox
            android:id="@+id/checkBox02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/acc2" />

        <CheckBox
            android:id="@+id/checkBox03"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/acc3" />

        <Button
            android:id="@+id/write_tag"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:minWidth="48dip"
            android:onClick="onRecordTag"
            android:text="@string/recordTag"
            android:textSize="20dp" />

        <Button
            android:id="@+id/save"
            style="?android:attr/buttonStyleSmall"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="onSave"
            android:text="@string/save"
            android:textSize="20sp" />

    </LinearLayout>
```

11.1.6 result.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:focusableInTouchMode="false"
        android:gravity="center"
        android:padding="10sp"
        android:text="@string/tag_list"
        android:textSize="25sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/list"
        android:layout_width="fill_parent"
        android:layout_height="1dp"
        android:focusableInTouchMode="false"
        android:gravity="center"
        android:padding="10sp"
        android:textSize="15sp"
        android:textStyle="bold" />

    <ScrollView
        android:id="@+id/ScrollView01"
        android:layout_width="fill_parent"
        android:layout_height="match_parent" >


    <TextView
        android:id="@+id/tag_permission_user"
        android:layout_width="fill_parent"
        android:layout_height="150dp"
        android:focusableInTouchMode="false"
        android:gravity="top|center_vertical"
        android:padding="10sp"
        android:textSize="15sp"
        android:textStyle="bold" />

    </ScrollView>

</LinearLayout>
```

11.1.7 tag_permissions.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="75dp"
        android:focusableInTouchMode="false"
        android:gravity="center"
        android:padding="10sp"
        android:text="@string/user_permissions"
        android:textSize="22sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/tag_permission_user"
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:text="TextView" />

    <CheckBox
        android:id="@+id/checkBox1tag"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:clickable="false"
        android:duplicateParentState="true"
        android:text="@string/acc1" />

    <CheckBox
        android:id="@+id/checkBox2tag"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/acc2" />

    <CheckBox
        android:id="@+id/checkBox3tag"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/acc3" />

</LinearLayout>
```

11.1.8 main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/action_settings"/>

     <item
        android:id="@id/list"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/show_list"/><item android:id="@+id/close"
android:orderInCategory="100" android:showAsAction="never" android:title="@string/close" />

</menu>
```